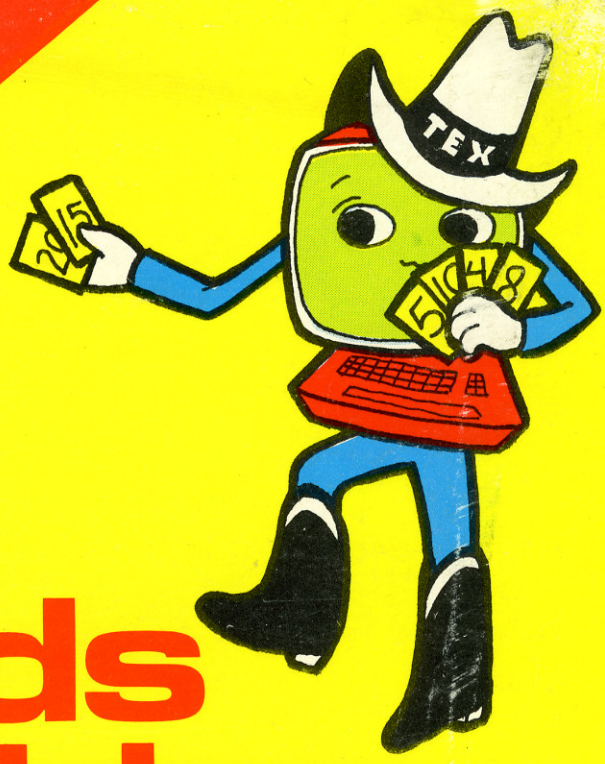
The Texas Instruments BASIC Manual



kids working with computers

Thomas Milton Kemnitz

Lynne Mass

TRILLIUM PRESS

KIDS WORKING WITH COMPUTERS

The Texas Instruments BASIC Manual

Thomas Milton Kemnitz

Lynne Mass

Trillium Press
New York

TABLE OF CONTENTS

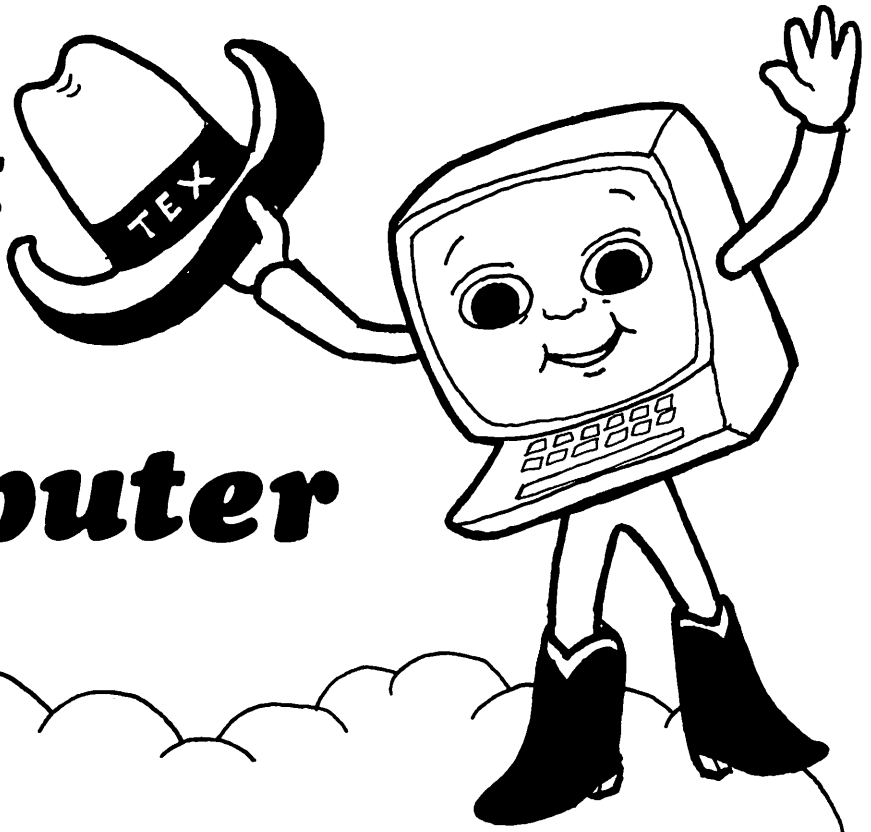
Trillium Press Inc.
Box 921
Madison Square Station
New York, N.Y. 10159

(212) 505-1441

ISBN: 0-89824-059-X
Printed in U.S.A.

Cover and illustrations by Lori Schlendorf.

meet the computer 1.



To Start:

1. Plug in the computer and turn on the monitor.
2. When the screen tells you, press any key to begin.
3. Press 1 for TI BASIC.
4. The screen will show:

   TI BASIC READY

   (The blinking black box is called the "cursor.")
5. Type CALL CLEAR and push ENTER.

You are now ready for your first program!

If you make a mistake, hold down FCTN and push S, and then type again.

Type EXACTLY what you see!

```
10 PRINT "HELLO, MY NAME IS (type your name)." *
20 PRINT "I AM LEARNING TO RUN THE TI 99/4A." *
30 PRINT "MY TEACHER IS (type in teacher's name)." *
40 PRINT "(make up something about yourself)." *
```

push ENTER

* Always remember to push ENTER at the end of each line.

Now that you have typed your program, type LIST and push ENTER. What happens?

Try typing RUN. Did you remember to push ENTER?

What happens if you type NEW ENTER and then RUN ENTER?

Review

What do these words do?

CALL CLEAR _____

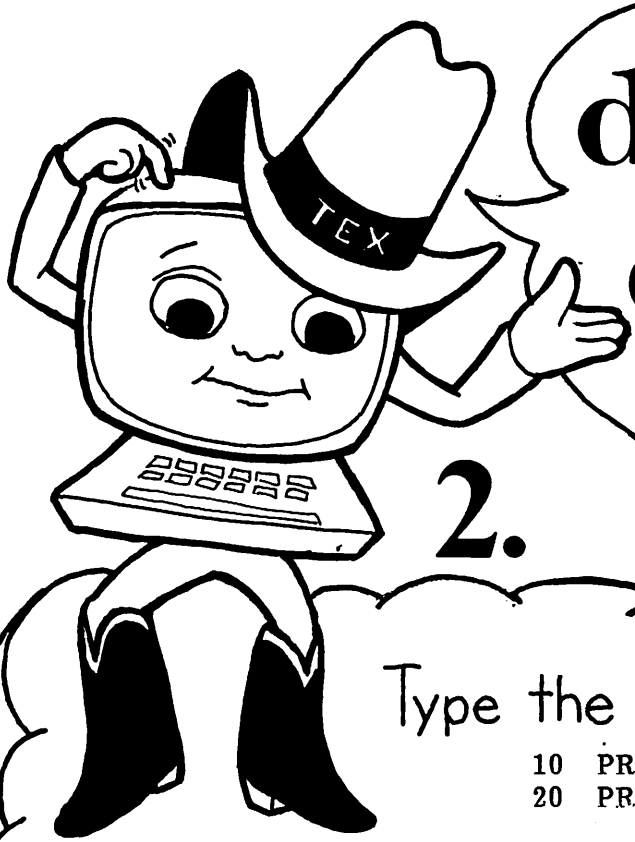
LIST _____

RUN _____

NEW _____

Why are these words called commands?

_____



do I need quotation marks?

2.

Type the following program...

```
10  PRINT "HI, MOM !"
20  PRINT "HI, DAD !"
```

(don't forget LIST and RUN .)

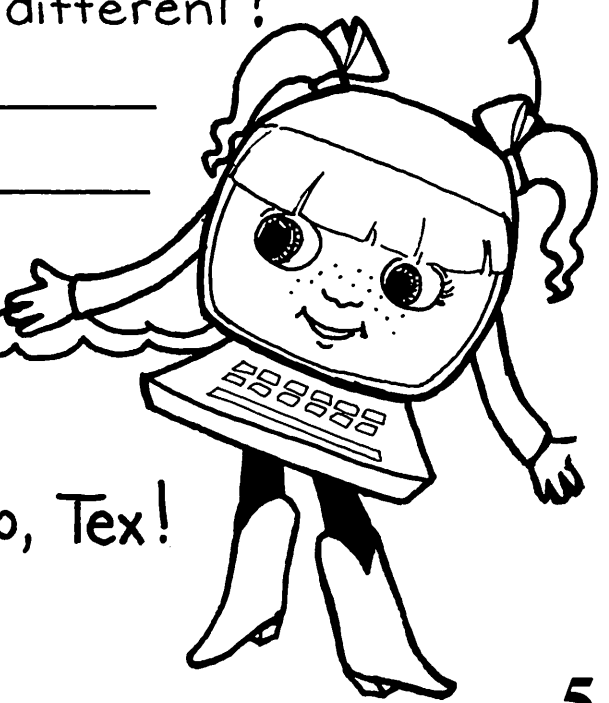
Type NEW . Now try the same words without quotation marks:

```
10  PRINT HI, MOM
20  PRINT HI, DAD
```

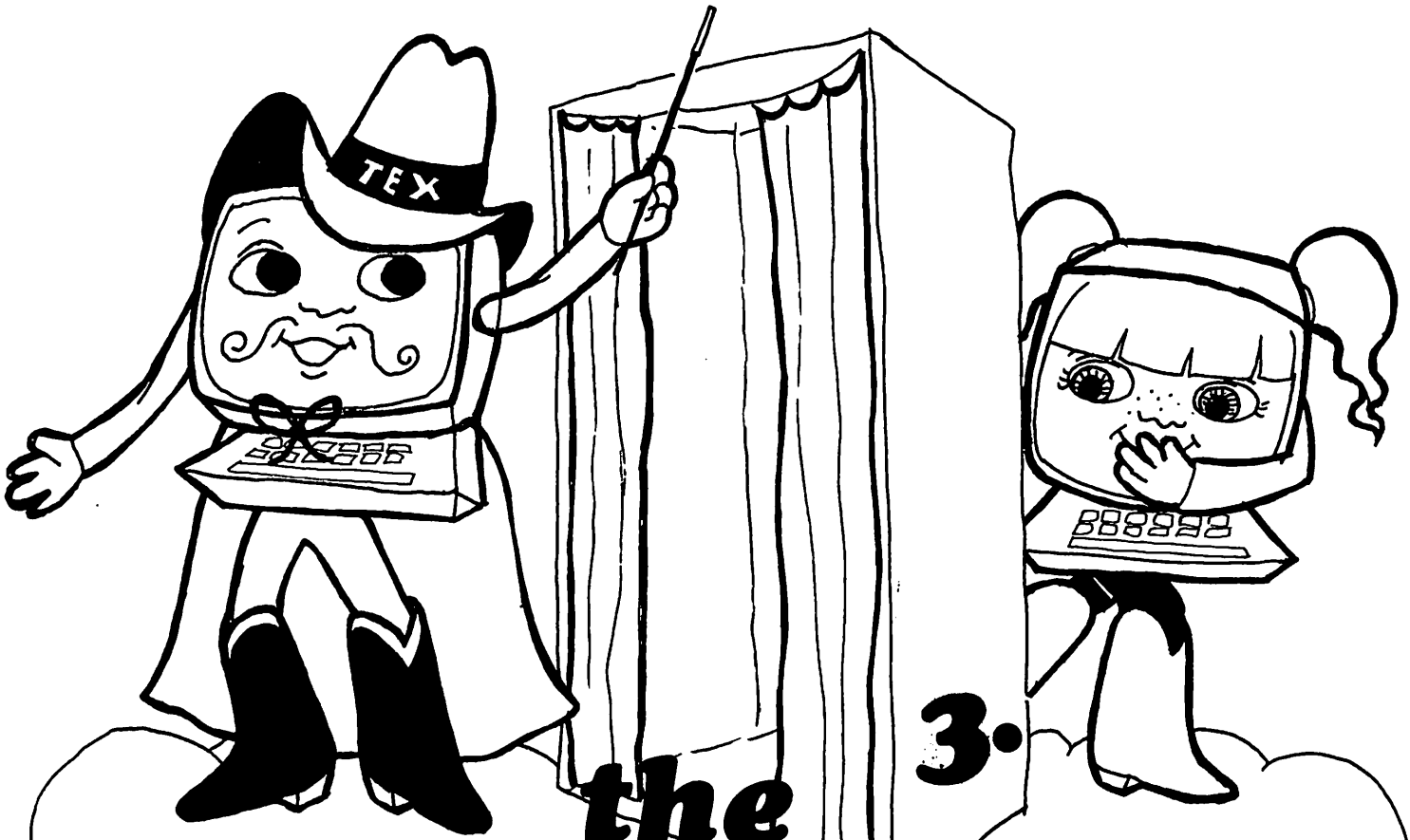
How are these two programs different?

_____

_____



Guess you do, Tex!



3. the disappearing act

Program the computer with :

```
10  PRINT "RUBBER BABY BUGGY
    BUMPERS."
```

Clear the screen by typing CALL CLEAR .

- Does this erase your program? How can you test this?

You can see what is in the memory by typing LIST .

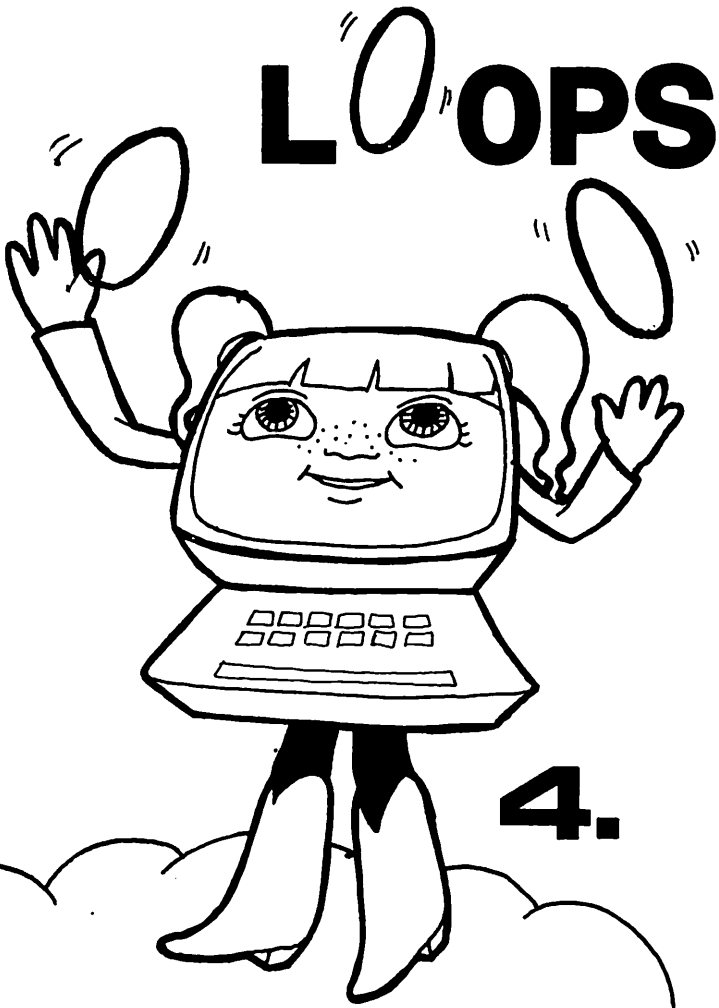
Now type in these commands:

```
NEW
CALL CLEAR
```

Does this clear the screen also?

Is your program erased?

computer LOOPS



4.

Type the following program and RUN it.

```
10  PRINT "HELP - MY COMPUTER
    WENT CRAZY"
20  GOTO 10
```

When you have seen enough, hold FCTN and press $ 4 .

- This is called a LOOP and is controlled by the GOTO command. Now try this program:

```
10  FOR X = 1 TO 5
20  PRINT "HELP - MY COMPUTER
    WENT CRAZY"
30  NEXT X
40  PRINT "NO, I HAVE IT
    UNDER CONTROL."
```

This is called a FOR-NEXT loop.

This statement says I am going to do something 5 times.

This statement is what will be done.

This statement sends the computer back to the counter.

Retype line 10:

```
10 FOR X = 1 TO 12
```

This will erase what was there before, like on a tape recorder.

What do you think will happen after you type RUN ? _____

Try typing it again with no space between the X and the = sign. RUN it. Did that make a difference?

_____

Try it with no space between 1 and To. Did that make a difference?

_____

Try it with no space between = and 1. Did that make a difference?

_____

Try it with no space between FOR and X. Did that make a difference?

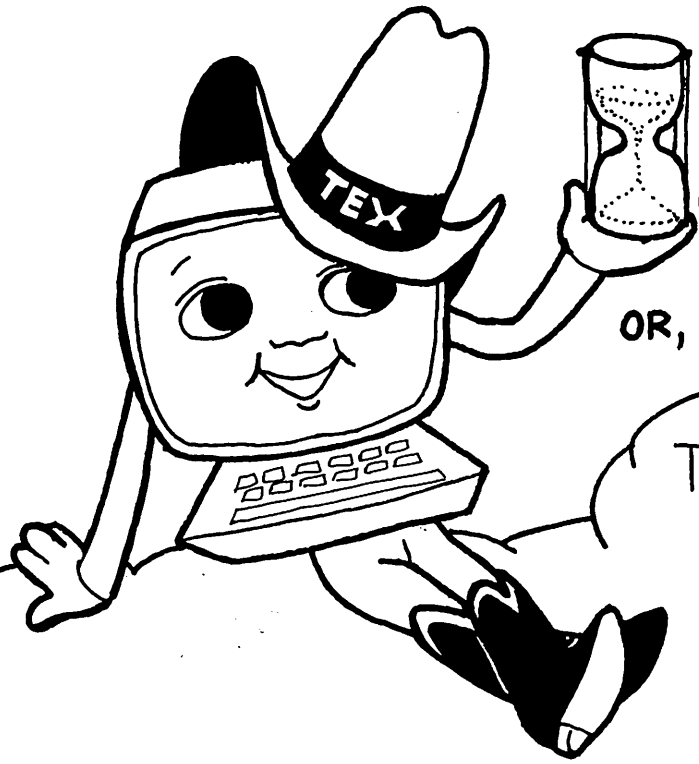
_____

Try it with no space between TO and 12. Did that make a difference?

_____

Try this program:

```
10  FOR B = 1  TO 588
20  PRINT "A";
30  NEXT B
40  PRINT "STOP!!!": "PHEW!!!!!!"
```



5. the egg timer

OR, MORE FOR-NEXT LOOPS

Type the following program:

```
2  CALL CLEAR
10  PRINT "WAIT RIGHT HERE !"
20  FOR X = 1 TO 3000
30  NEXT X
40  PRINT "YOUR TIME IS UP."
```

Did time go by between the printing of WAIT RIGHT HERE and YOUR TIME IS UP?_____

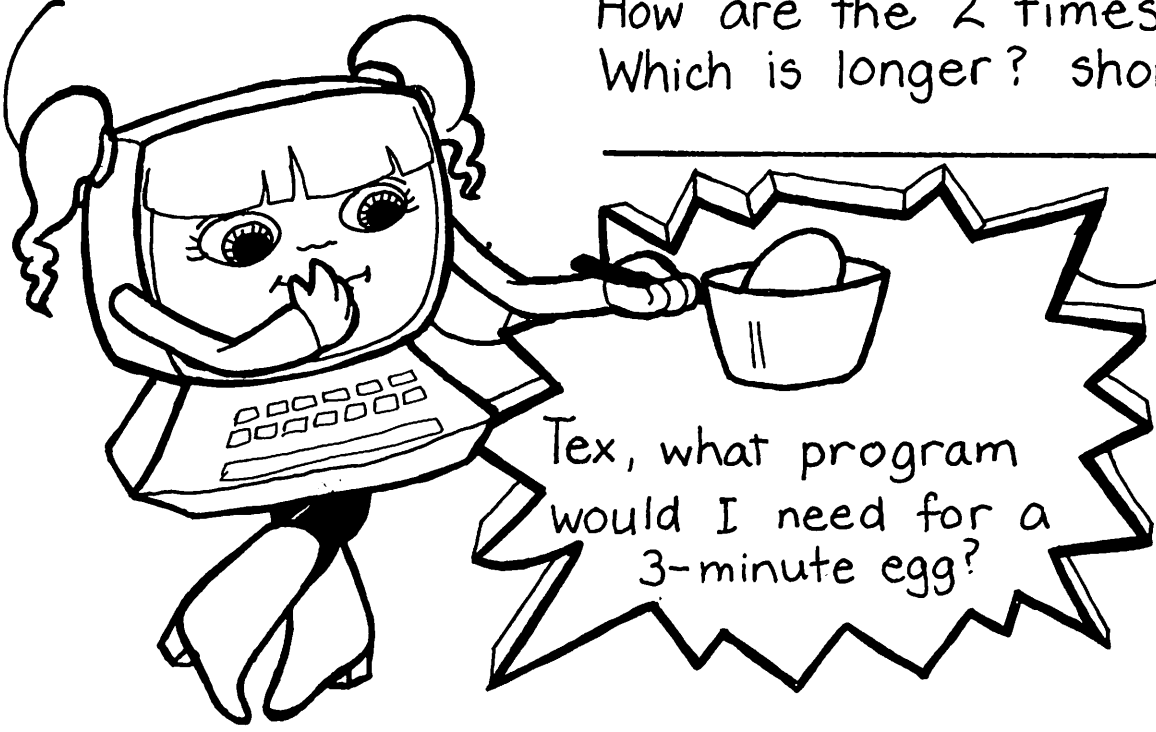
How much time went by? _____

Now change line 20 as follows and RUN it: `20 FOR X = 1 TO 10000`

then change it to: `20 FOR X = 1 TO 2000`

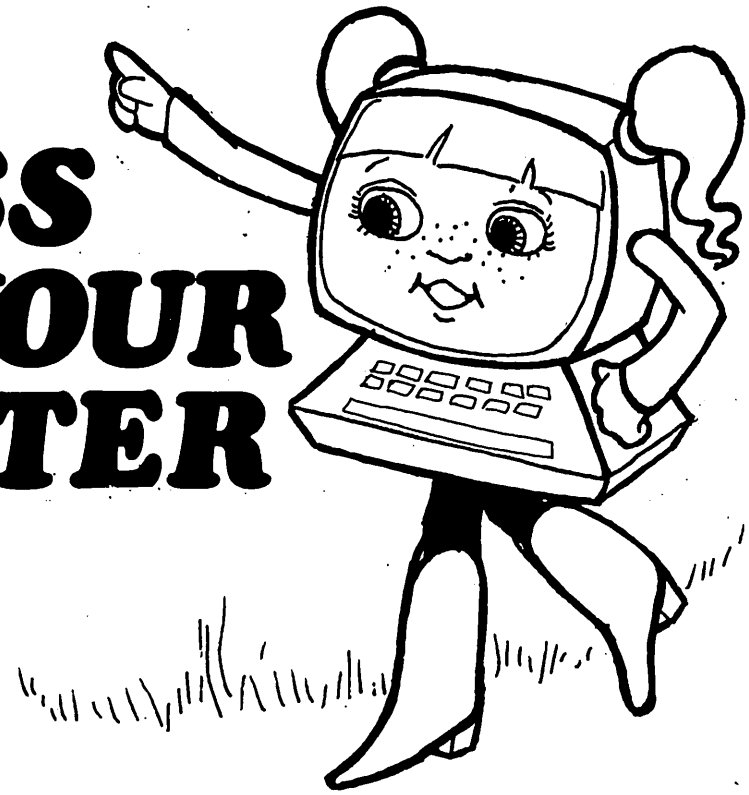
How are the 2 times different? Which is longer? shorter?

_____



Tex, what program would I need for a 3-minute egg?

6. BOSS AROUND YOUR COMPUTER



Type the following program:

```
2  CALL CLEAR
10  FOR X = 1 TO 50
20  PRINT "NOW I AM PROGRAMMING."
30  PRINT "I AM STILL AT IT."
40  PRINT  "AND I CAN'T STOP."
50  NEXT X
```

What happens? _____

_____

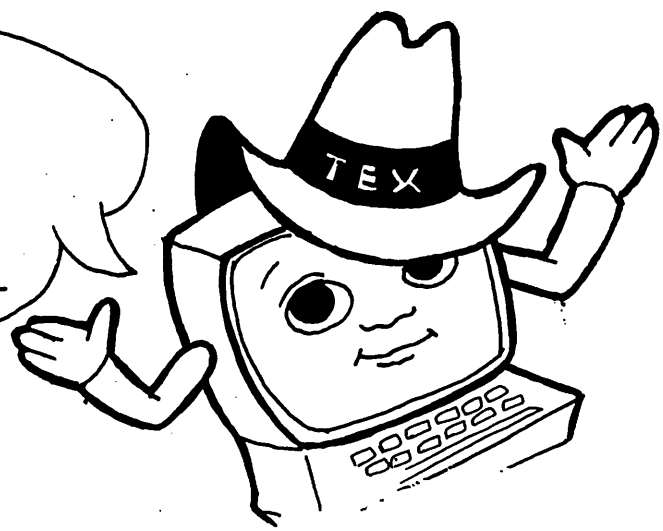
How many times do you think you told the computer to write lines 20, 30, and 40?

_____

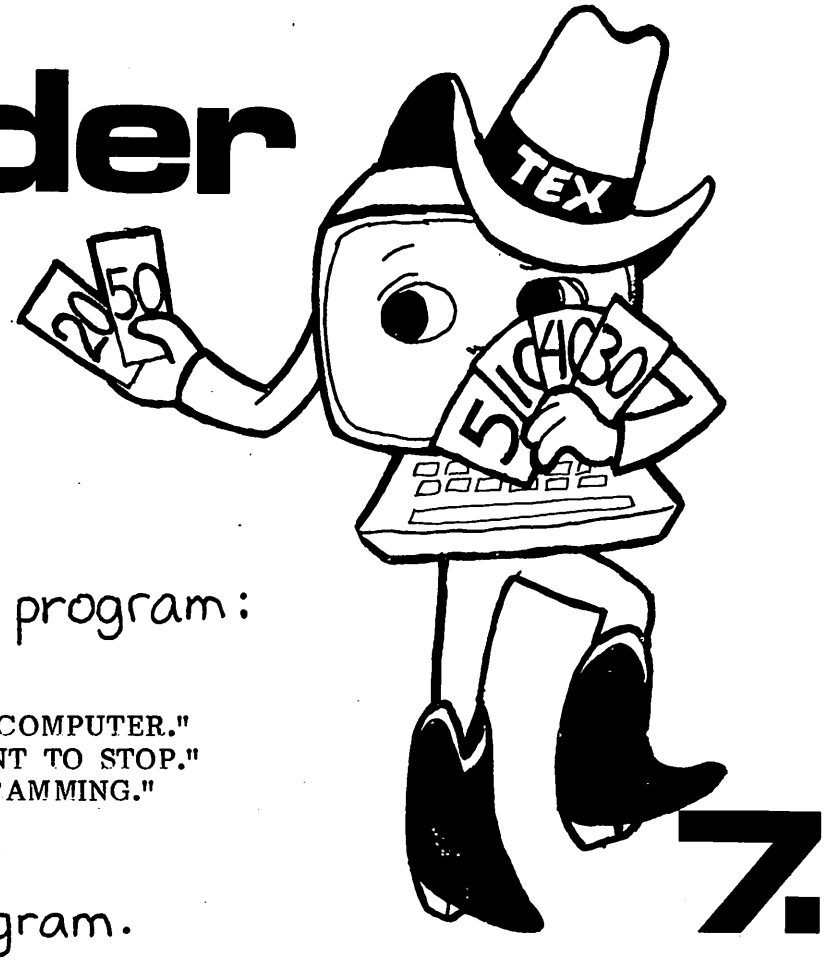
Keep the same program and add 45 PRINT: PRINT
Does the program look different? _____

Do I have to retype the whole program to add a line?

_____



line order



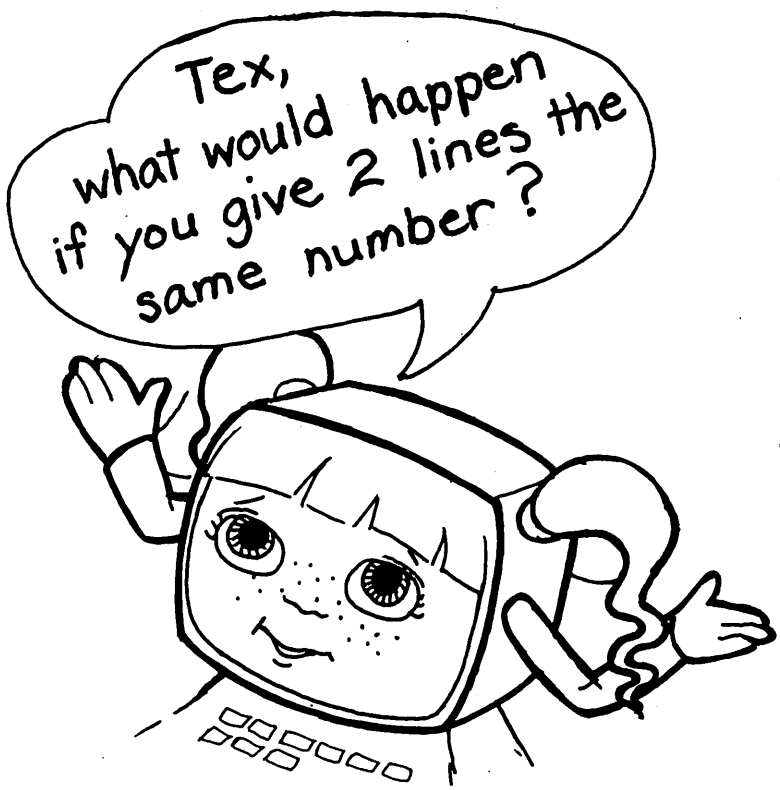
7.

Type the following program:

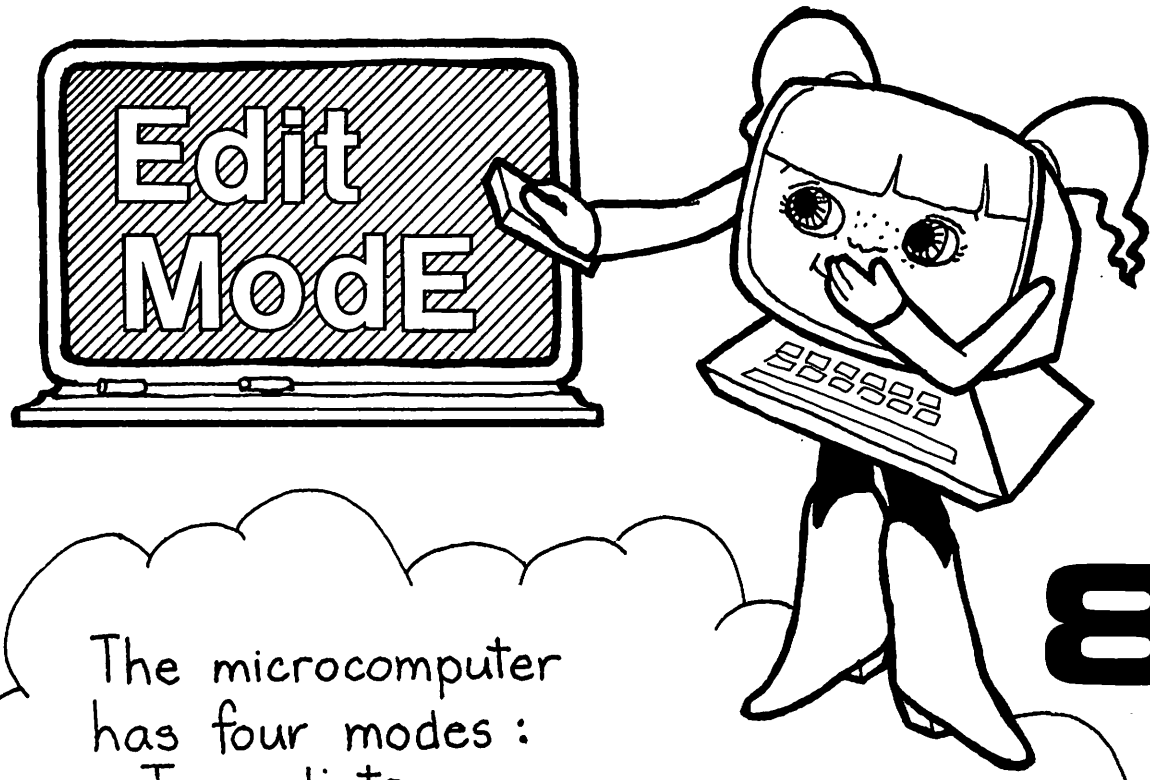
```
10  PRINT "I AM USING THE COMPUTER."
20  PRINT "AND I DON'T WANT TO STOP."
5   PRINT "NOW I AM PROGRAMMING."
```

Now LIST the program.

What did the computer do for you?



Can you figure out a way to add more lines between the beginning and the end?



8.

The microcomputer has four modes:

1. Immediate
2. Programming
3. Execution
4. Edit

The IMMEDIATE mode is when you tell the computer `PRINT 5 + 7 + 3` and push [ENTER] and it prints `15`. In the immediate mode you give the computer a command and it does it. You do not use line numbers.

In the PROGRAMMING mode you use line numbers. The computer just stores your instructions.

In the EXECUTION mode, you tell the computer `RUN`, `LIST`, `PRINT`, etc., and it does whatever you command.

In the EDIT mode, you can change a line or more very easily.

Suppose you typed in

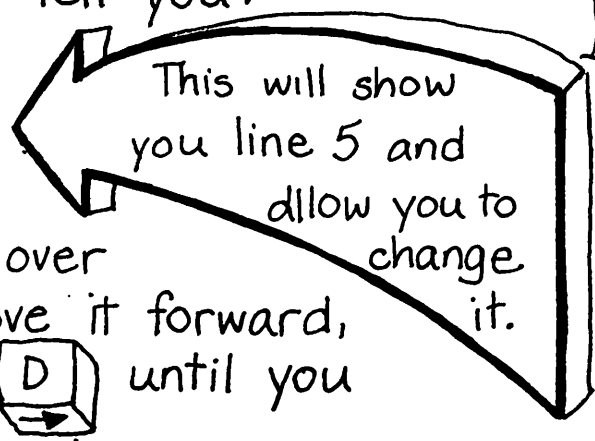
```
5 CALL DEAR
10 PLINT "LOVE"
```

and tried to RUN it.

The computer will tell you:

```
BAD NAME IN 5.
```

Type in EDIT 5



The cursor will blink over the C in CALL. To move it forward, hold down FCTN and D until you get to the "D" in "DEAR." Now you can change DEAR to CLEAR.

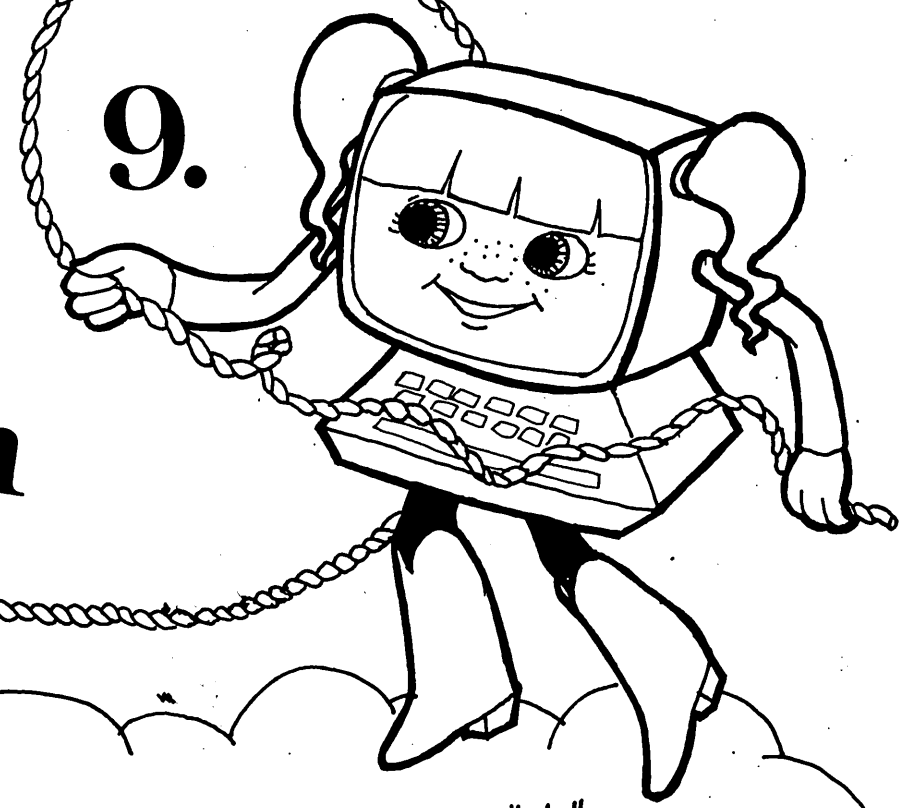
Now ENTER and RUN .

The computer will tell you

```
INCORRECT STATEMENT IN 10.
```

You can now type in EDIT 10 and make your change in line 10!

9. String A Design



A string ($) – a variable and a "$" – is one or more characters.

A string can be interesting! For example, try this:

```
02  CALL CLEAR
05  A$ = "GOOD MORNING"

10  PRINT A$
```

What happens? _____

You can even get a number value for a string. LEN is a code that will tell you how many characters – in this case, letters and spaces – are in the string.

Try this:

```
10  PRINT LEN  (A$), LEN ("YES")
```

What do the 12 and 3 represent?
(CLUE: **LEN** is a computer word for "length".)

Now try this:

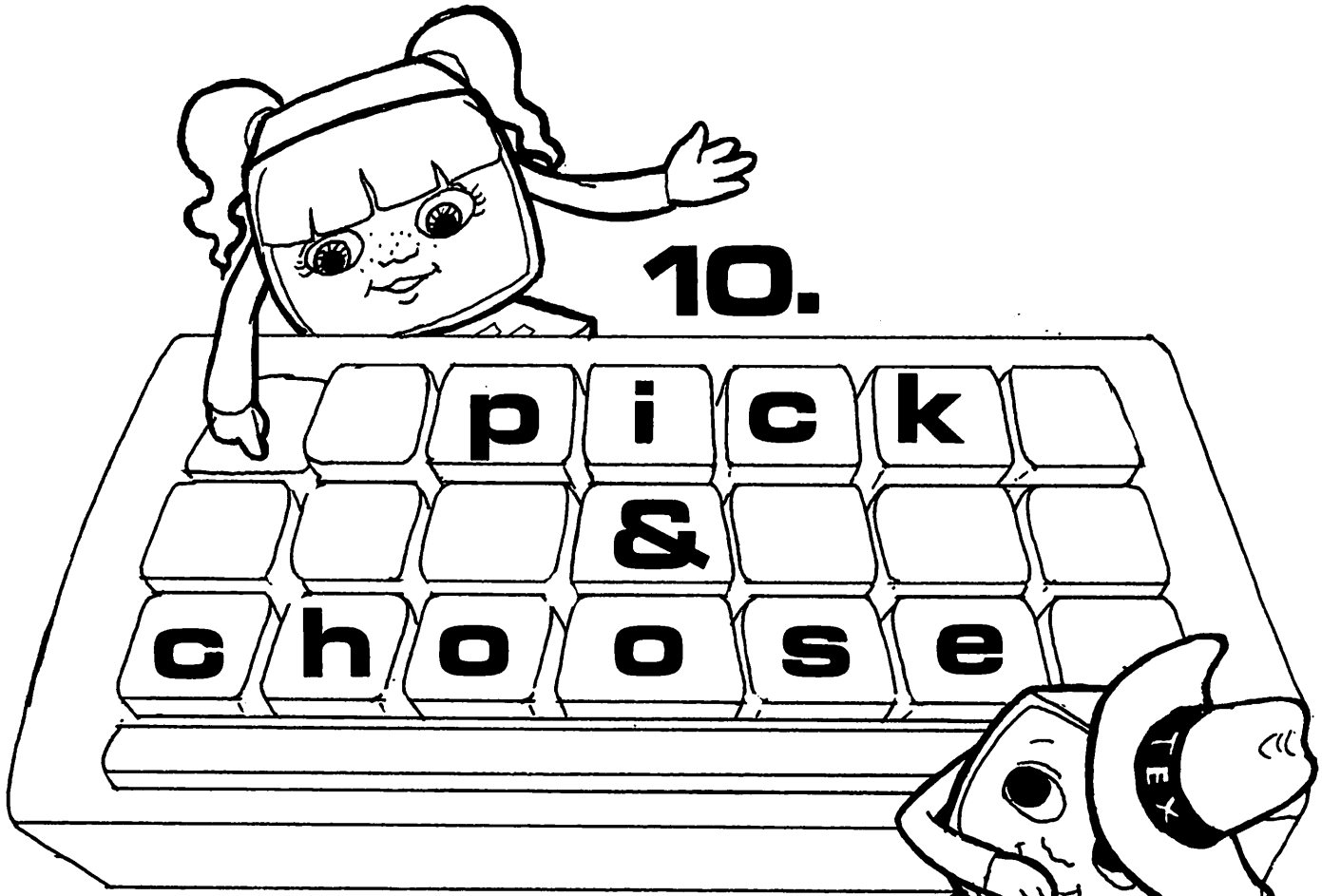
```
10  PRINT SEG$ (A$,1,4)
```

What do you get? It's good if you see GOOD .
How do you think you could get the computer to print MORNING ?

Now try this:

```
10  FOR N = 1 TO LEN (A$)
20  PRINT SEG$ (A$,1,N)
30  NEXT N
```

Isn't that cute?
Why does this happen? Since A$ has 12 characters, this loop will be done 12 times with N= 1, 2, 3, ... 11, 12. The first time only, the 1st character will be printed (and N=1). The second time, the 1st two characters will be printed (and N=2), and so on.



Here is a program to try with a friend:

```
 2  CALL CLEAR
 5  PRINT "TYPE IN YOUR AGE."
10  INPUT A
12  CALL CLEAR
15  PRINT "YOUR AGE IS "
20  PRINT A
RUN
```

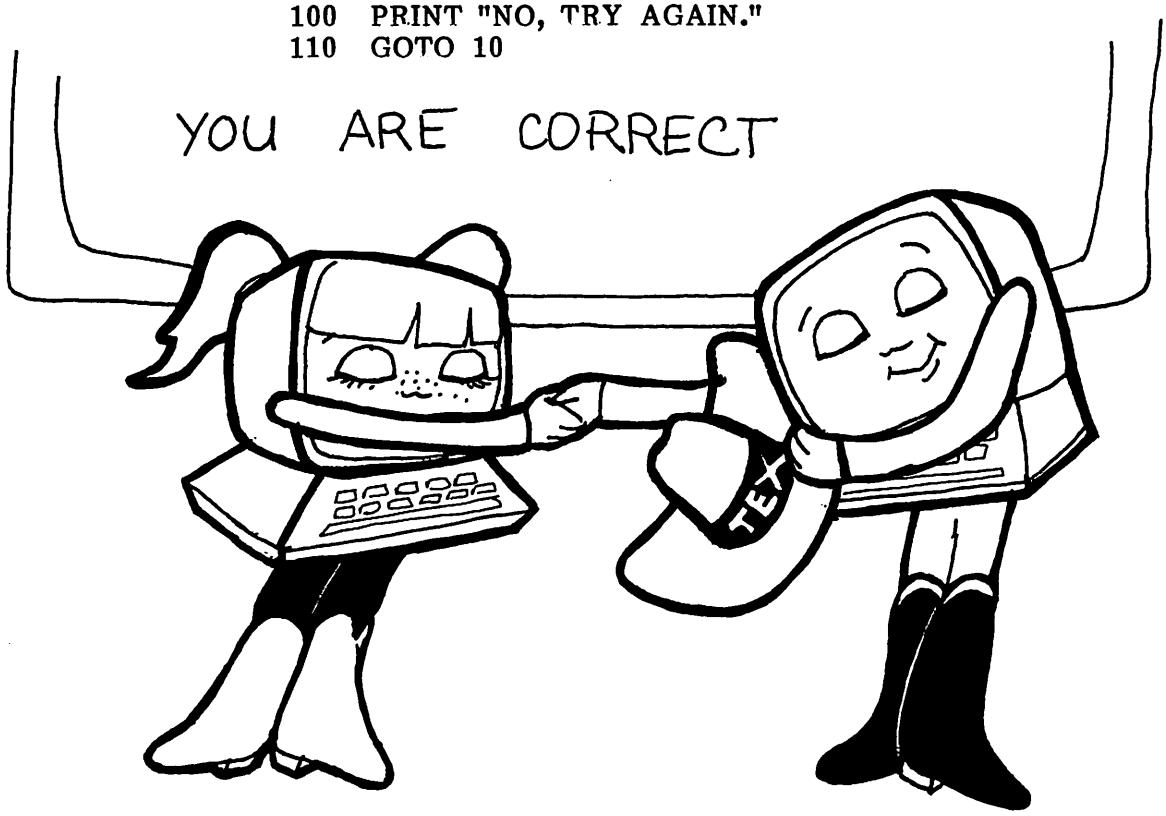
INPUT asks your friend to type in a number while the program is running.

When you are done with this program, type NEW.

Now that you have learned to construct a program with an INPUT, you can take advantage of computer logic to construct games. The computer is able to decide if two numbers are equal or if one number is greater (>) or lesser (<) than another. If they are equal, you can tell the computer to do one thing; if unequal, you can tell it to do something else.

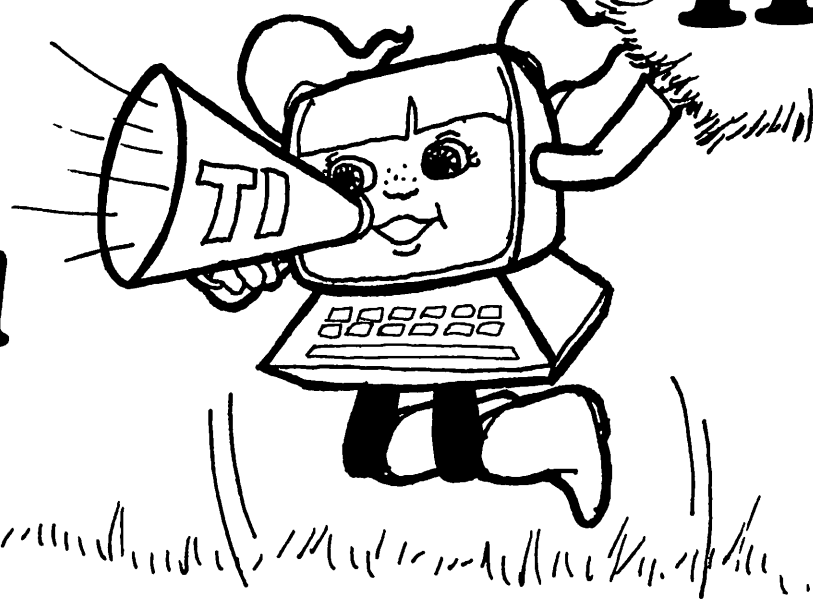
Try this simple game:

```
2  CALL CLEAR
5  PRINT "PICK A NUMBER.  TYPE IT
   IN.  THE CHOICES ARE
   1,2, OR 3."
10  INPUT N
20  IF N<>3 THEN 100
30  IF N = 3 THEN 40
40  PRINT "YOU ARE CORRECT."
50  END
100  PRINT "NO, TRY AGAIN."
110  GOTO 10
```



11.

GIVE ME A T!



Use "Easy Guessing Game" program from last lesson.
Change line 30 to:

```
30  IF N = 3 THEN 200
```

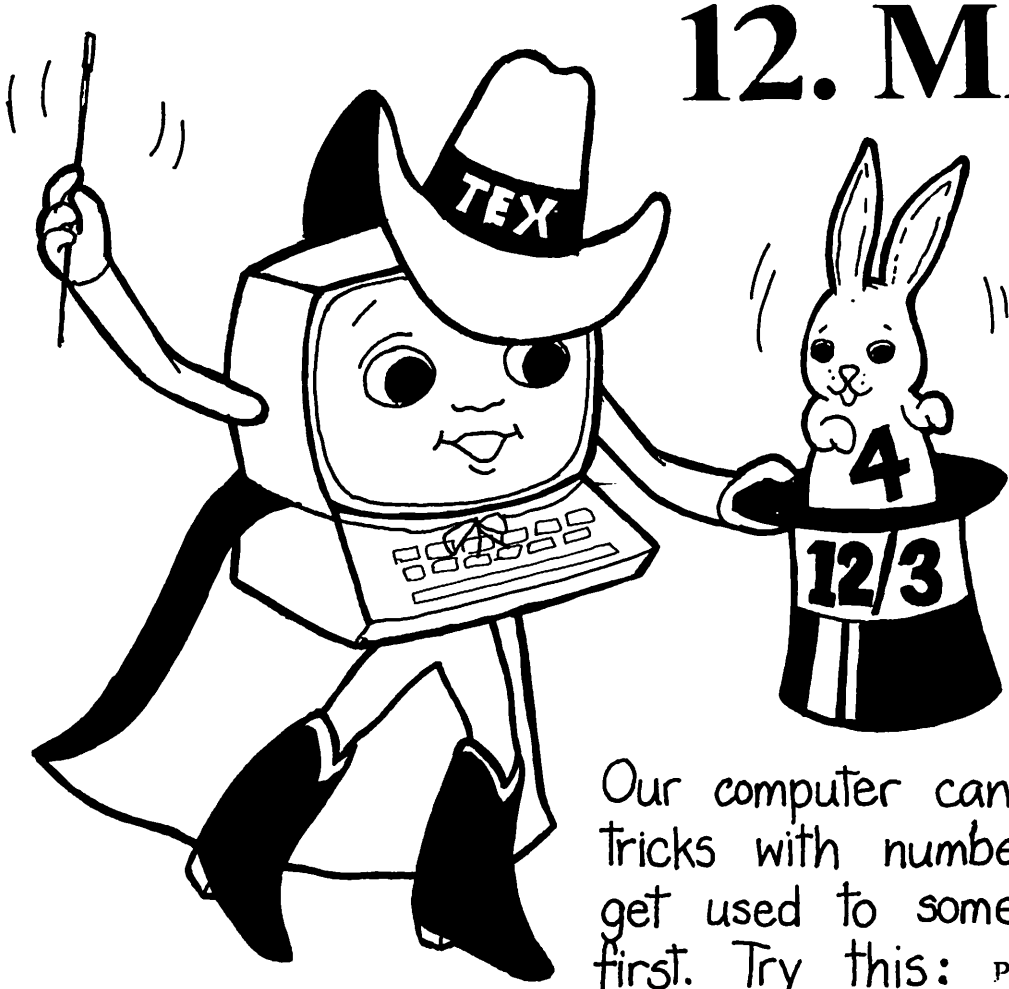
Now we are going to add sound!

CALL SOUND is the command that calls up sound.

```
200  ZAP = 110
210  FOR BEEPS = 1 TO 10
220  CALL SOUND (-500, ZAP, 1)
230  ZAP = ZAP + 110
240  NEXT BEEPS
250  GOTO 40
```

- What happens if you type HONKS instead of BEEPS in lines 210 and 240? _____

12. MATH MAGIC



Our computer can do many tricks with numbers. Let's get used to some easy ones first. Try this: PRINT 3 + 4

The TI-99/4A can do six different arithmetic operations:

1. ADDITION (+)	PRINT 5 + 7
2. SUBTRACTION (–)	PRINT 6-2
3. MULTIPLICATION (*)	PRINT 7*8
4. DIVISION (/)	PRINT 63/7
FRACTIONS (/)	PRINT 3/2
5. EXPONENTIATION	PRINT 4*4*4*4*4
OR...	PRINT 4^5
6. SQUARE ROOT	PRINT SQR (16)

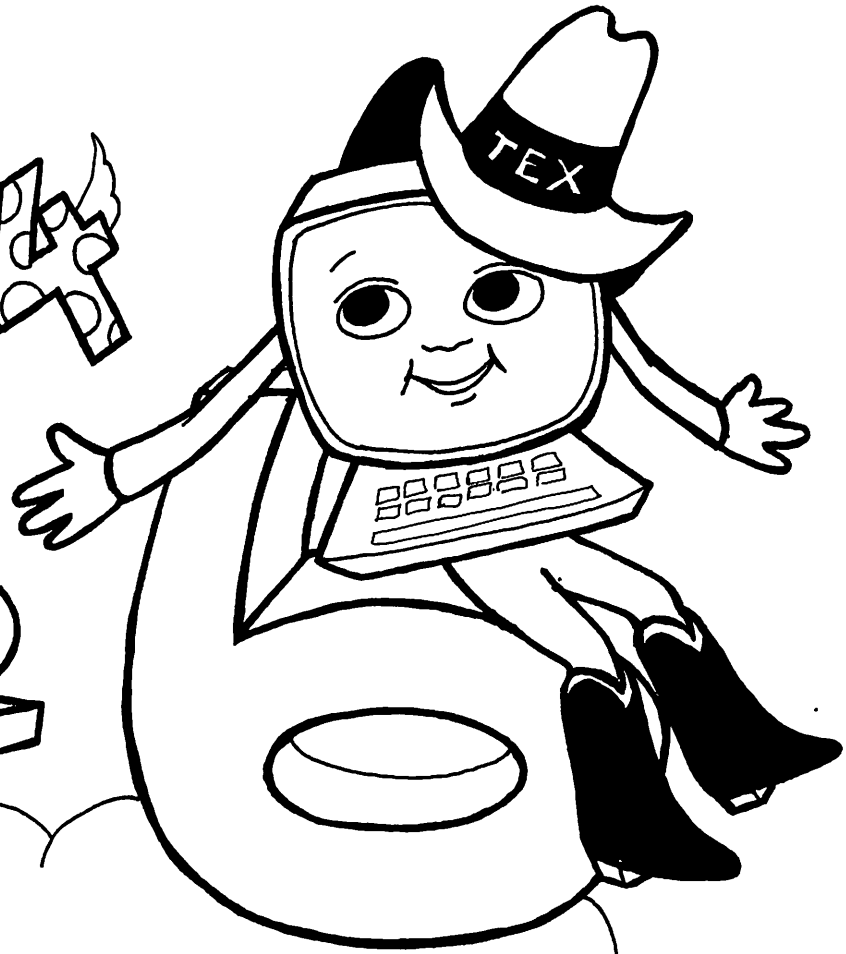
Can you make up some math problems of your own?

_____

_____

_____

13. Some Strange Numbers



Type the following: PRINT .37542905716742

What do you see?

The computer rounds off to what number of places?

Is that always true?

Try 375.42905716742. What does the computer do to it?

To get an idea of what computers do with numbers in expressions, do these in your head or on paper first...

and then on the computer.

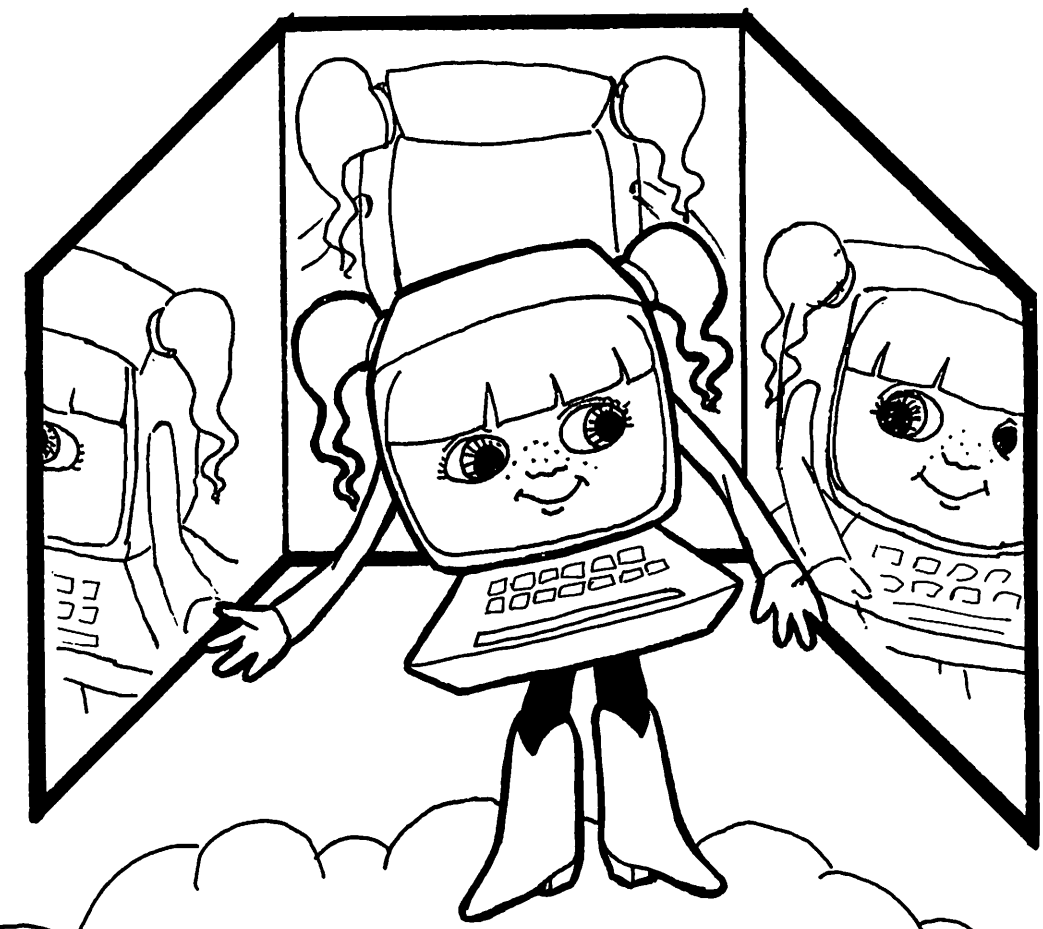
1. 3 + 1 * 2	_____	9. SQR (9) - SQR (5)	_____
2. 2 + 4/2	_____	10. 5 * 8 + SQR (9)	_____
3. 3 + 2 + 6	_____	11. 3 - 2 * 25	_____
4. 7 - 3 + 2	_____	12. 3^2 + 7^2 * 2^2 - 3	_____
5. 3 + 2^2 + 4 * 2	_____	13. SQR (25) - SQR (16) * 3^2	_____
6. 3 + 2^3 - 2 * 2 + 5	_____	14. 4 + 2^2/2^3 - 1	_____
7. 3 + 6 - 2 + 4^2	_____	15. 7/3 + 4	_____
8. 8 + 4/3 - 3	_____	16. SQR (81) * 11 + 2	_____

The computer does math operations in the following order:

1. All operations within parentheses. If one set is inside another, it does the inside set first.
2. Exponents and square roots.
3. Multiplication and division.
4. Addition and subtraction.

Now, using brackets, how could you write these expressions more clearly?

repeat



14.

One advantage of computers is their ability to perform repetitive tasks. Follow these programs involving square roots and see the shortcuts that develop! Try this program:

```
10  PRINT 1, SQR (1)
20  PRINT 2, SQR (2)
30  PRINT 3, SQR (3)
40  PRINT 4, SQR (4)
50  PRINT 5, SQR (5)
60  PRINT 6, SQR (6)
70  PRINT 7, SQR (7)
80  PRINT 8, SQR (8)
90  PRINT 9, SQR (9)
100  PRINT 10, SQR (10)
```

- SQR is computer language for "square root" and SQR(x) is the way you write "square root of a number" where "X" is the number.

This is a shortened form of the same program. Try it!

Type NEW

```
02  CALL CLEAR
10  N = 1
20  PRINT N, SQR (N)
30  N = N + 1
40  IF N < = 10 THEN 20
```

How does this 4-line program compare with the 10-line program you ran first?

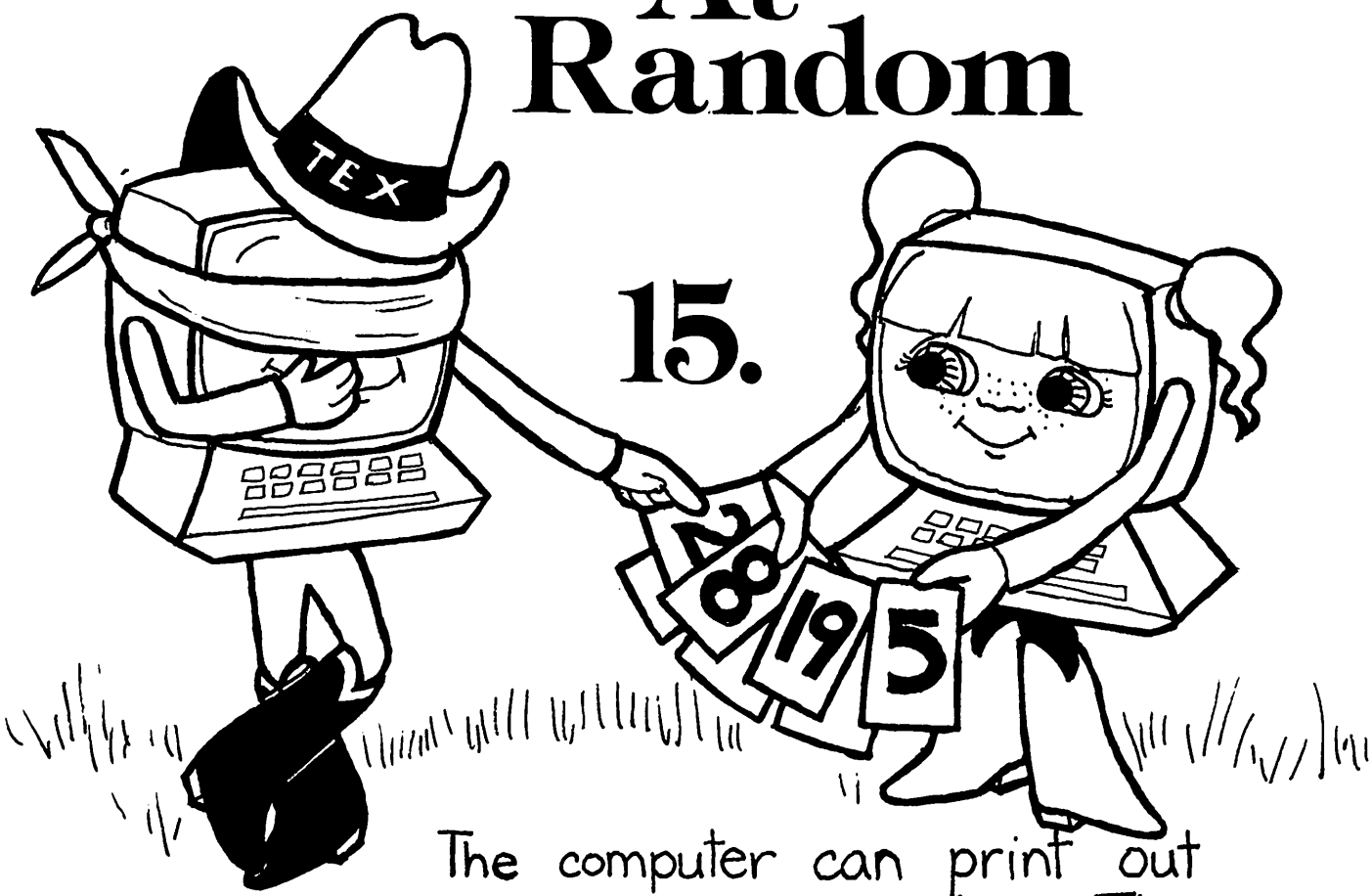
Using the "loop" above, we can shorten the program even more ... Type NEW

```
10  FOR N = 1 TO 10
20  PRINT N, SQR (N)
30  NEXT N
```

Can you figure out a program for printing a table of square roots for only the even integers from 10 to 20?

At Random

15.



The computer can print out numbers which are not predictable. These form the basis for many computer games - the numbers are formed randomly and have a code - RND - for random numbers.

To get random numbers on the TI-99/4A, you have to have the word RANDOMIZE in the program.

To see what (RND+0) does, try this!

```
10 RANDOMIZE
20 PRINT (RND+0)
```

What did you get?

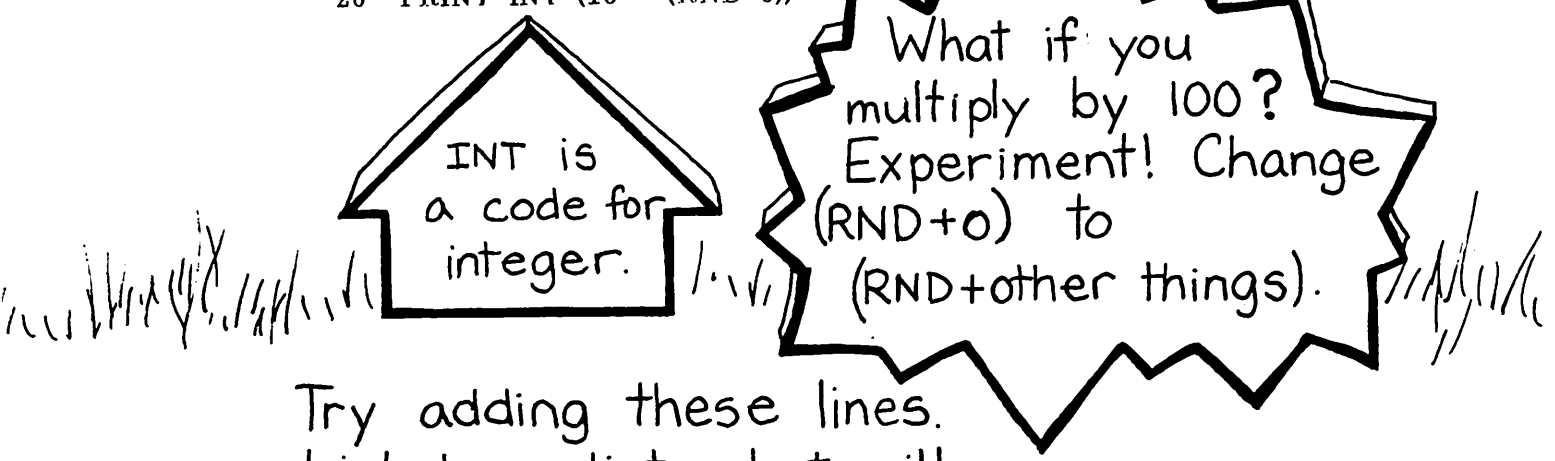
Try it again - did you get the same number? Are the numbers whole numbers or decimals?

How can you get a number > 1 (greater than one)? Try this!

```
20  PRINT 10 * (RND+0)
```

To get rid of the numbers after the decimal, try this:

```
20  PRINT INT (10 * (RND+0))
```



Try adding these lines. List & predict what will happen before you run it.

```
2  CALL CLEAR
15  FOR N= 1 TO 10
30  NEXT N
```

Change lines 15 and 20 to:

```
15  FOR N= 1 TO 90
20  PRINT INT (10 * (RND+0));
```

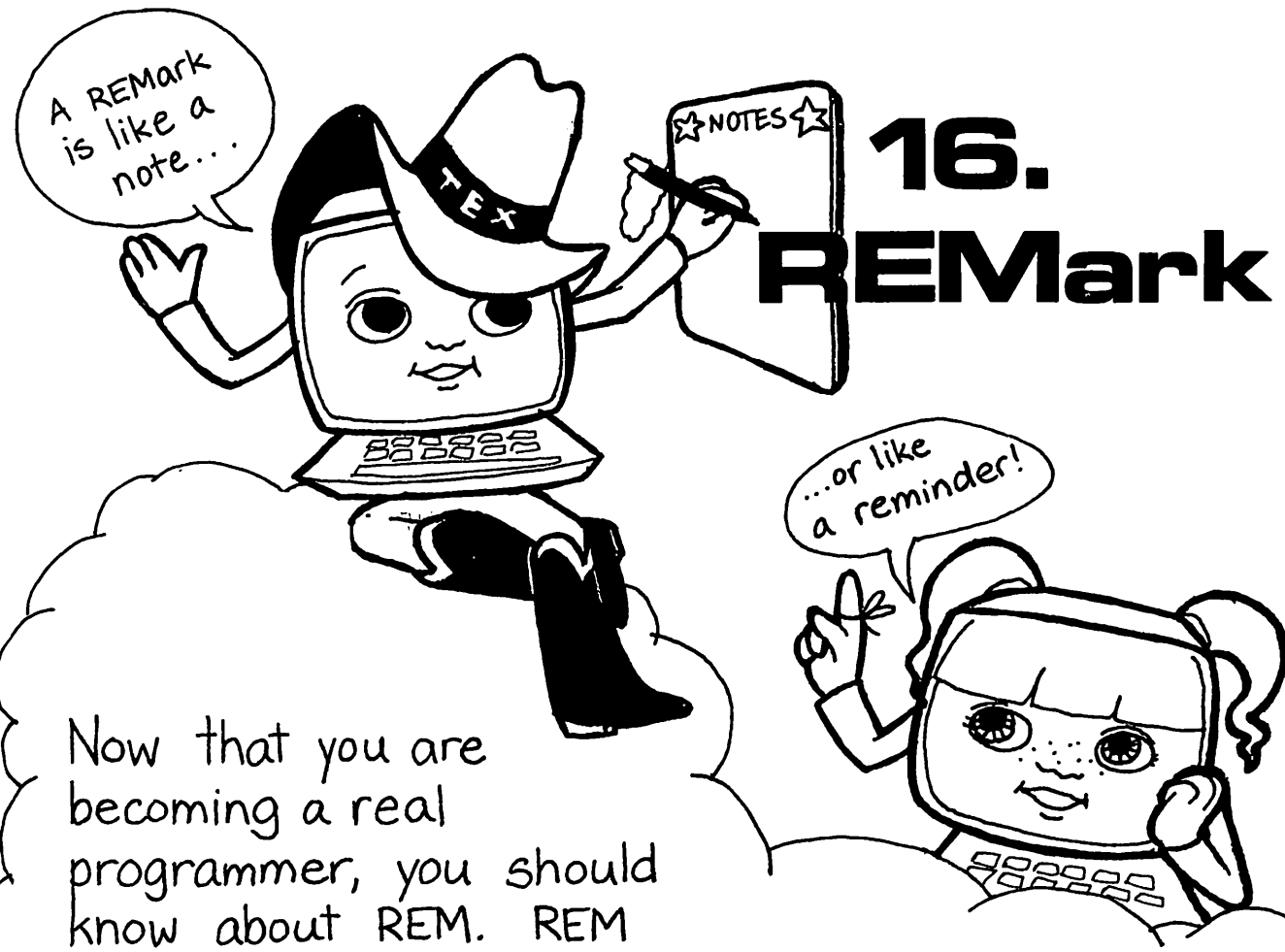
What happens? _____

_____

What does the ; do? _____

_____

16. REMark



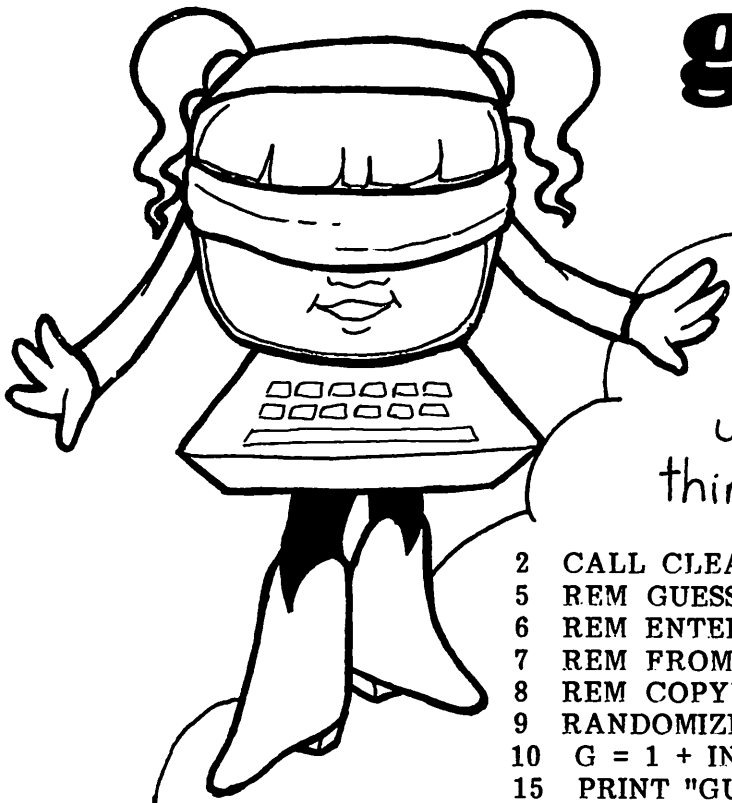
Now that you are becoming a real programmer, you should know about REM. REM is short for REMARK. Any line in a program that is headed by REM will not RUN, but it will LIST. So, a REMARK is like a note right in the middle of the program.

Programmers use REM to:

1. Name programs.
2. Remind themselves of bugs or changes.
3. Date programs.
4. Note programming language.
5. Identify variables.
6. Remind themselves or tell others what they expect a part of the program to do.

From now on we'll use REM in some of our programs!

17. make a guessing game



Here is a number game using RND and other things you have learned...

```
 2  CALL CLEAR
 5  REM GUESSING GAME
 6  REM ENTERED BY
 7  REM FROM KIDS WORKING WITH COMPUTERS
 8  REM COPYRIGHT 1983 TRILLIUM PRESS
 9  RANDOMIZE
10  G = 1 + INT (100 * (RND+0))
15  PRINT "GUESS A NUMBER FROM 1 TO 100."
20  LET C = 0
30  INPUT A
50  C = C + 1
60  IF A = G THEN 130
70  IF A < G THEN 100
80  PRINT "TOO HIGH"
90  GOTO 110
100  PRINT "TOO LOW"
110  GOTO 30
130  ZAP = 110
131  FOR BEEPS= 1 TO 10
132  CALL SOUND (-500, ZAP, 1)
133  ZAP = ZAP + 110
134  NEXT BEEPS
135  PRINT "VERY GOOD, YOU GUESSED IT!"
140  PRINT "YOU TOOK";C;"GUESSES."
```

Make up your own guessing game! Solve it on a disk or on the printer.

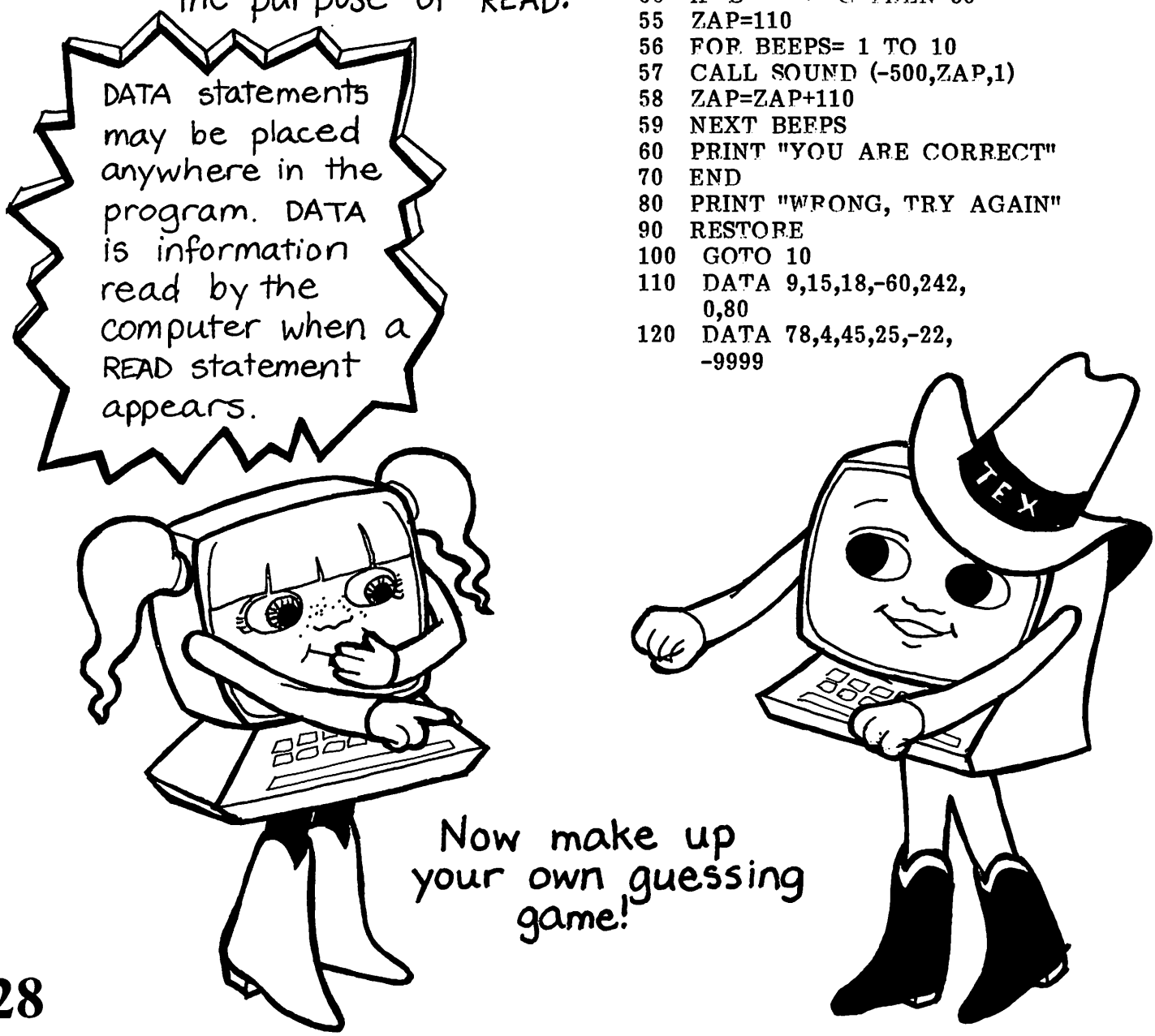
18. read data

The purpose of this program is to play a game in which you try to guess one of the numbers in lines 110 and 120 (these are called DATA STATEMENTS).

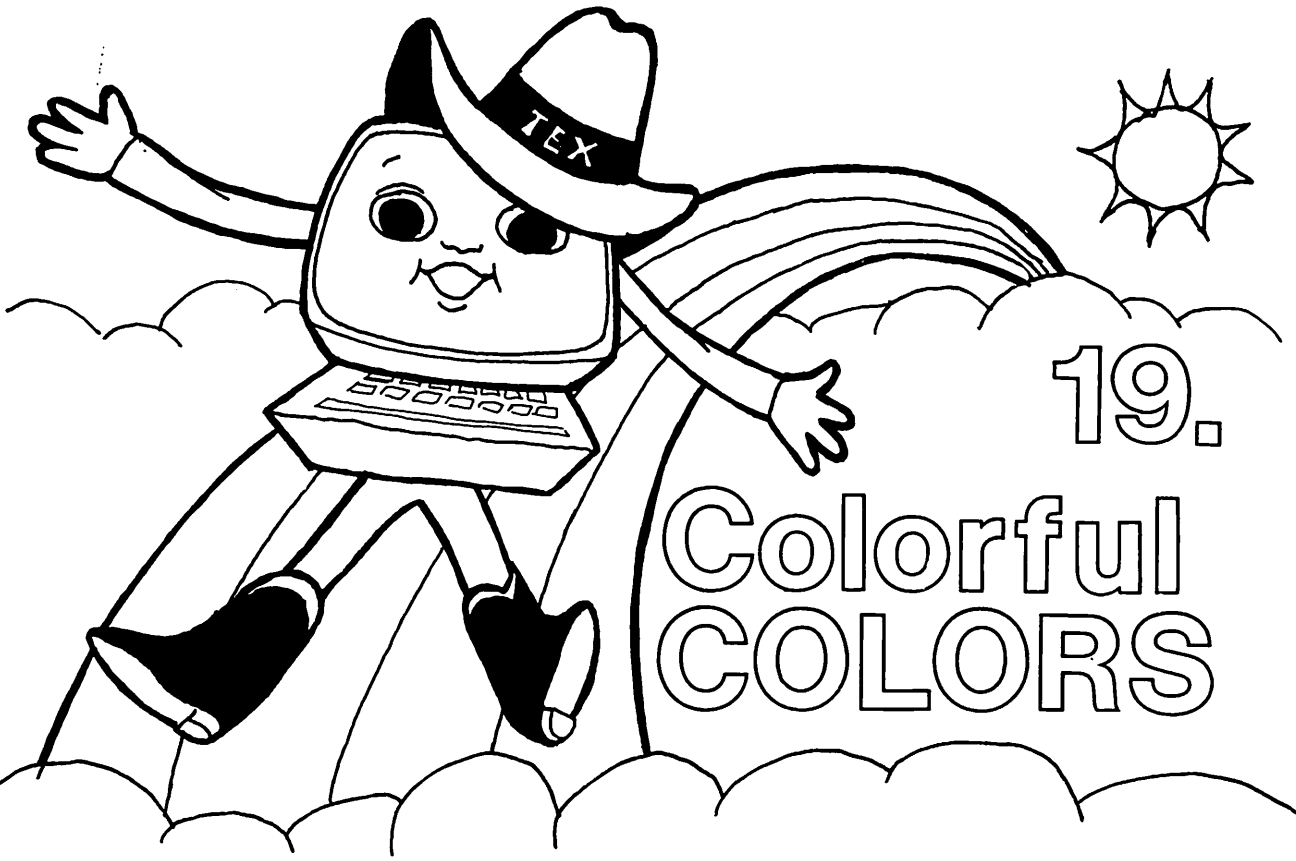
As you type in the program, try to figure out the purpose of READ.

```
2   CALL CLEAR
5   REM PICK A NUMBER
6   REM ENTERED BY (YOUR NAME)
7   REM DATE (TYPE TODAY'S DATE)
10  PRINT "PICK A NUMBER"
20  INPUT G
30  READ D
40  IF D =  - 9999 THEN 80
50  IF D < > G THEN 30
55  ZAP=110
56  FOR BEEPS= 1 TO 10
57  CALL SOUND (-500,ZAP,1)
58  ZAP=ZAP+110
59  NEXT BEEPS
60  PRINT "YOU ARE CORRECT"
70  END
80  PRINT "WRONG, TRY AGAIN"
90  RESTORE
100  GOTO 10
110  DATA 9,15,18,-60,242,
     0,80
120  DATA 78,4,45,25,-22,
     -9999
```

DATA statements may be placed anywhere in the program. DATA is information read by the computer when a READ statement appears.



Now make up your own guessing game!



19. Colorful COLORS

To get color and graphics (pictures made up of dots of color) on the screen, CALL COLOR.

Each color has a code number. For instance, dark red is 7.

You also have to tell the computer where you want the color. For instance, CALL SCREEN turns the background a color without changing the letters. Try:

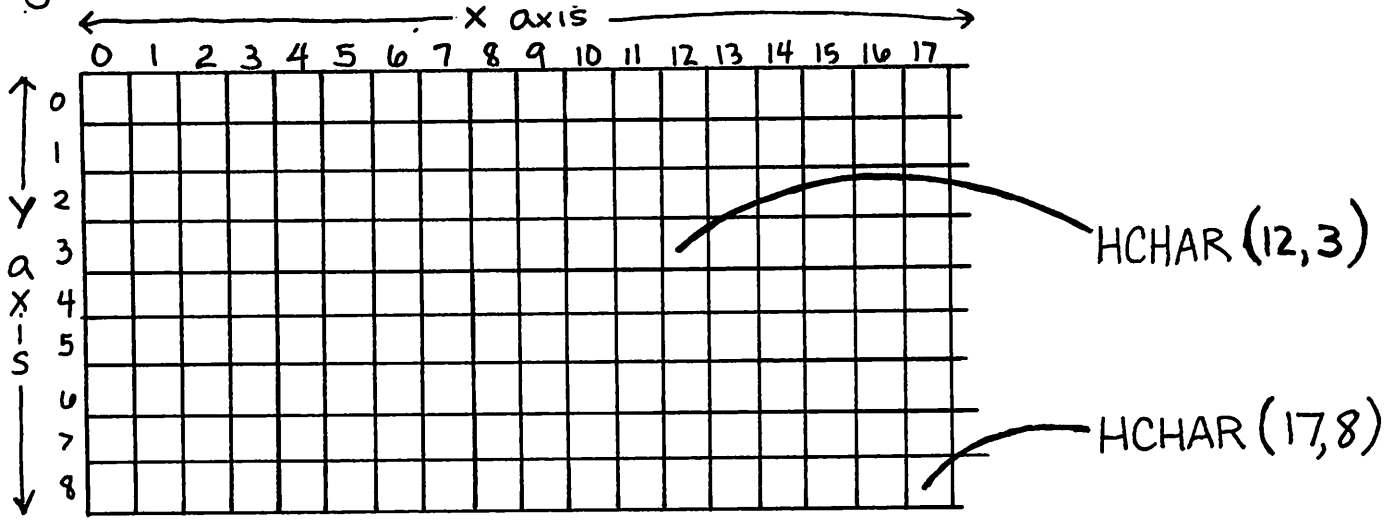
```
10 CALL SCREEN (7)
20 GOTO 10
```

Want a blinking screen that changes color? Try adding:

```
5  CALL SCREEN (5)
6  FOR X = 1 TO 1000
7  NEXT X
11 FOR X = 1 TO 1000
12 NEXT X
20 GOTO 5
```

When you've seen enough, hold down FCTN and press 4.

You can locate spots on a screen by using a grid like this:



To command the computer where to put a character, you look at the "address"- where the square is located, and use HCHAR or VCHAR. If you are going to put in only one character, it does not matter which you use. But, if you want to put a lot of characters in a line, use HCHAR (which stands for Horizontal CHARacters - in a line). If you want a lot up & down, use VCHAR (which stands for Vertical CHARacters — up and down. Try this

```
10 CALL CLEAR
20 CALL VCHAR (10,15,42)
```

Want to see it blink? Add

```
30 GOTO 10
```

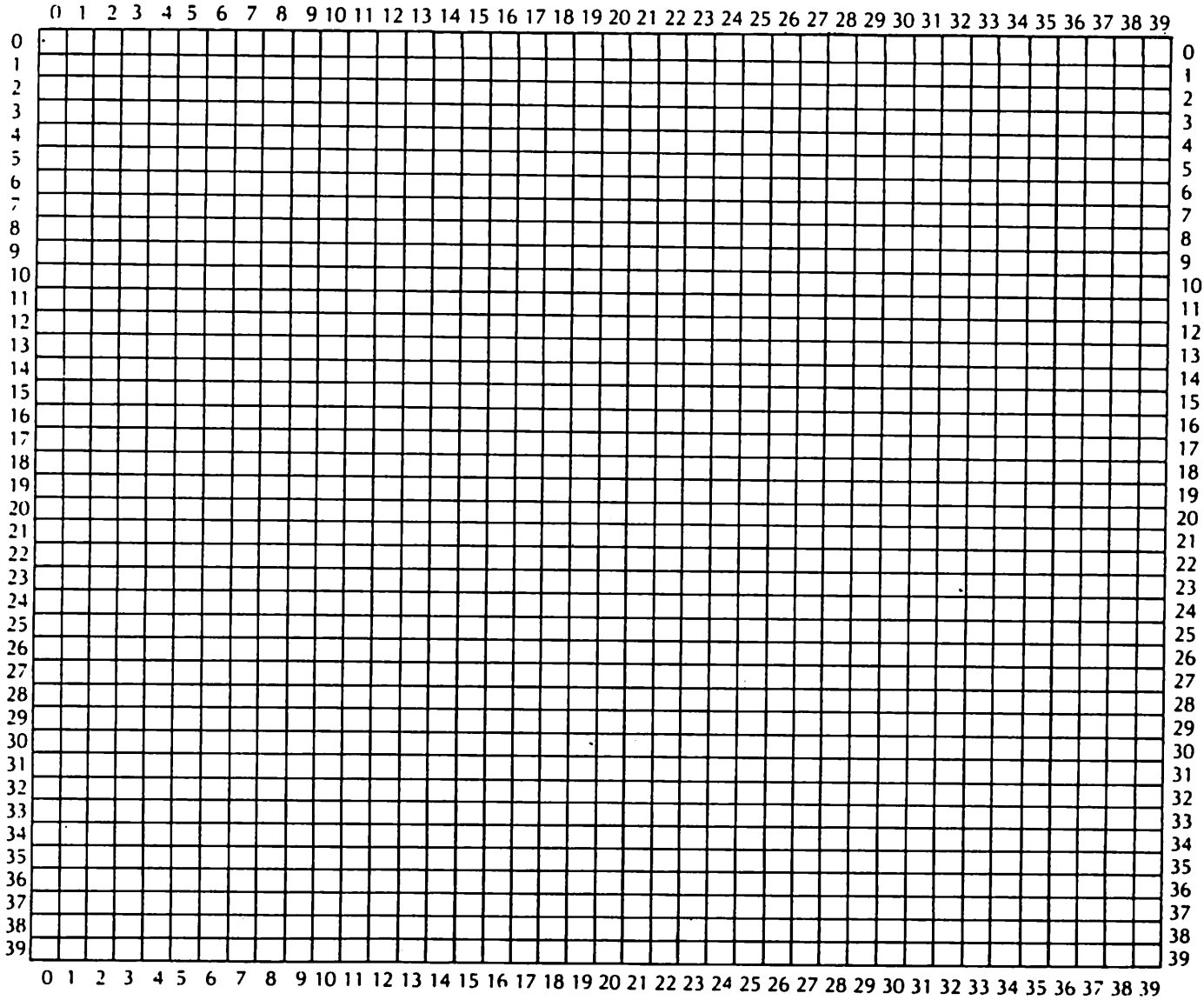
10 means 10 down on the Y axis, 15 means 15 across on the X axis. 42 is the number for an *!

- Look at the full grid on the next page. Can you write a program which would put an * in the middle of the screen and make it blink once per second?

Here are the colors and their numbers:

1. TRANSPARENT	7 DARK RED	13 DARK GREEN
2 BLACK	8 CYAN	14 MAGENTA
3 MEDIUM GREEN	9 MEDIUM RED	15 GRAY
4 LIGHT GREEN	10 LIGHT RED	16 WHITE
5 DARK BLUE	11 DARK YELLOW	
6 LIGHT BLUE	12 LIGHT YELLOW	

Practice plotting designs using the full grid below:



20. going straight

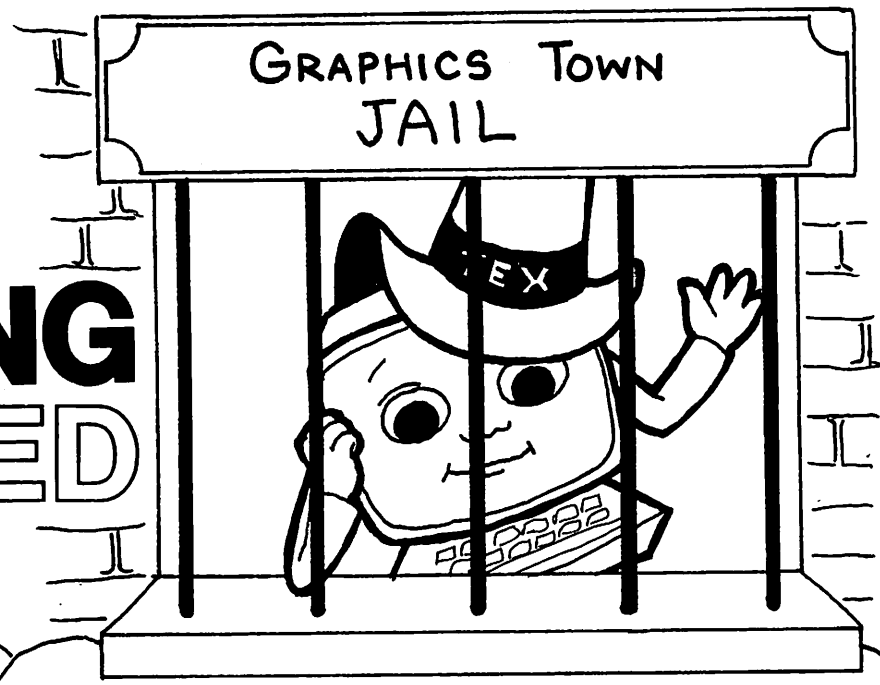
Suppose you want a straight line. There has to be an easier way than putting one little asterisk after another. If you want a horizontal line, use CALL HCHAR

```
2  REM GOING STRAIGHT
5  CALL CLEAR
10  CALL HCHAR (8,3,42,17)
```

8 means 8 down on the Y axis
3 means begin at 3 on X axis
42 means asterisk
17 means end at 17 on X axis

To draw a vertical line, edit line 10 by changing the H to V.

21 GETTING FRAMED

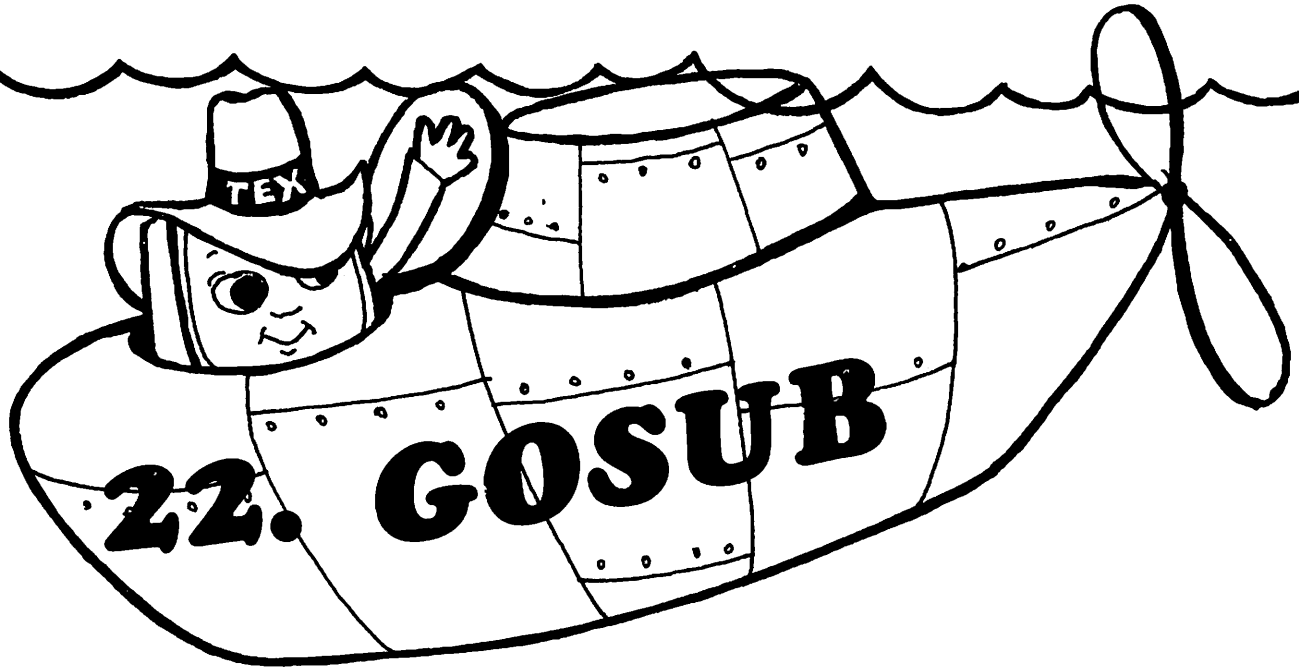


Now let's try drawing more than one line. In fact, let's make a frame!

Here's how:

```
5  CALL CLEAR
10  CALL VCHAR (1,3,42,22)
20  GOTO 100
100  CALL HCHAR (1,3,42,28)
110  GOTO 200
200  CALL VCHAR (1,30,42,22)
210  GOTO 300
300  CALL HCHAR (23,3,42,28)
310  GOTO 10
```

Now, can you figure out how to write your name in the frame? Try it!



22. GOSUB

GOSUB is a command that allows us to establish a subroutine which we can use many times in a program. GOSUB must always appear with the command RETURN If we wanted to add sound at the end of each line in drawing our frame, here is how we would do it.

```
 15  GOSUB  800
105  GOSUB  800
205  GOSUB  800
305  GOSUB  800
800  REM SOUND SUBROUTINE
810  ZAP = 110
820  FOR BEEPS = 1 TO 10
830  CALL SOUND (-500, ZAP,1)
840  ZAP = ZAP + 110
850  NEXT BEEPS
860  RETURN
```

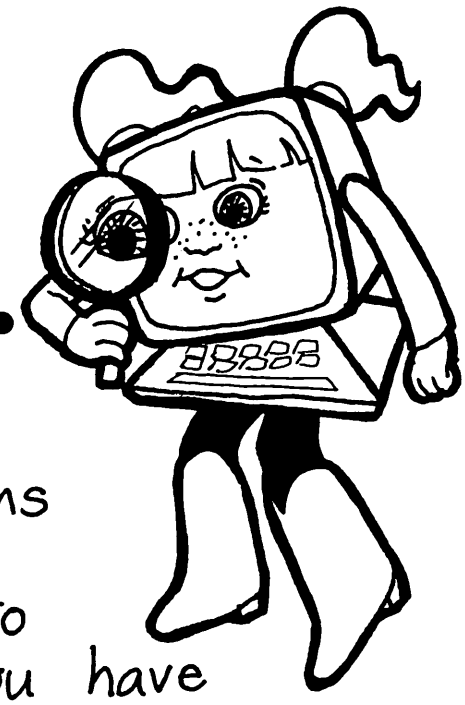
RUN it.

What happens if you change line 310 to:

310 GOTO 5 ? To

310 GOTO 310 ?

trace 23.



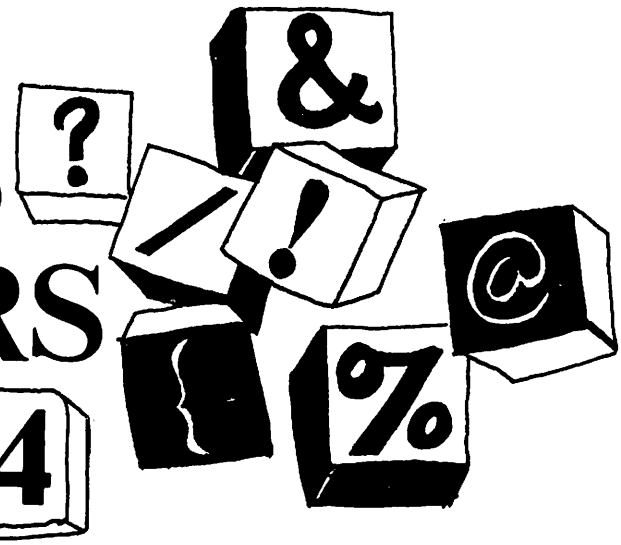
When you have subroutines, your programs can get very complicated. However, there is a way to follow all the steps that you have directed the computer to trace. in the order that it takes them. To see this, change the program we used in the last lesson by typing `310 END`

Now type TRACE
Press [ENTER], then [RUN].

All the numbers below the line were necessary for the computer to draw the last line and produce the music. Why do you suppose we see 830, 840 and 850 ten times?

Want to get rid of TRACE ?
Just type UNTRACE !

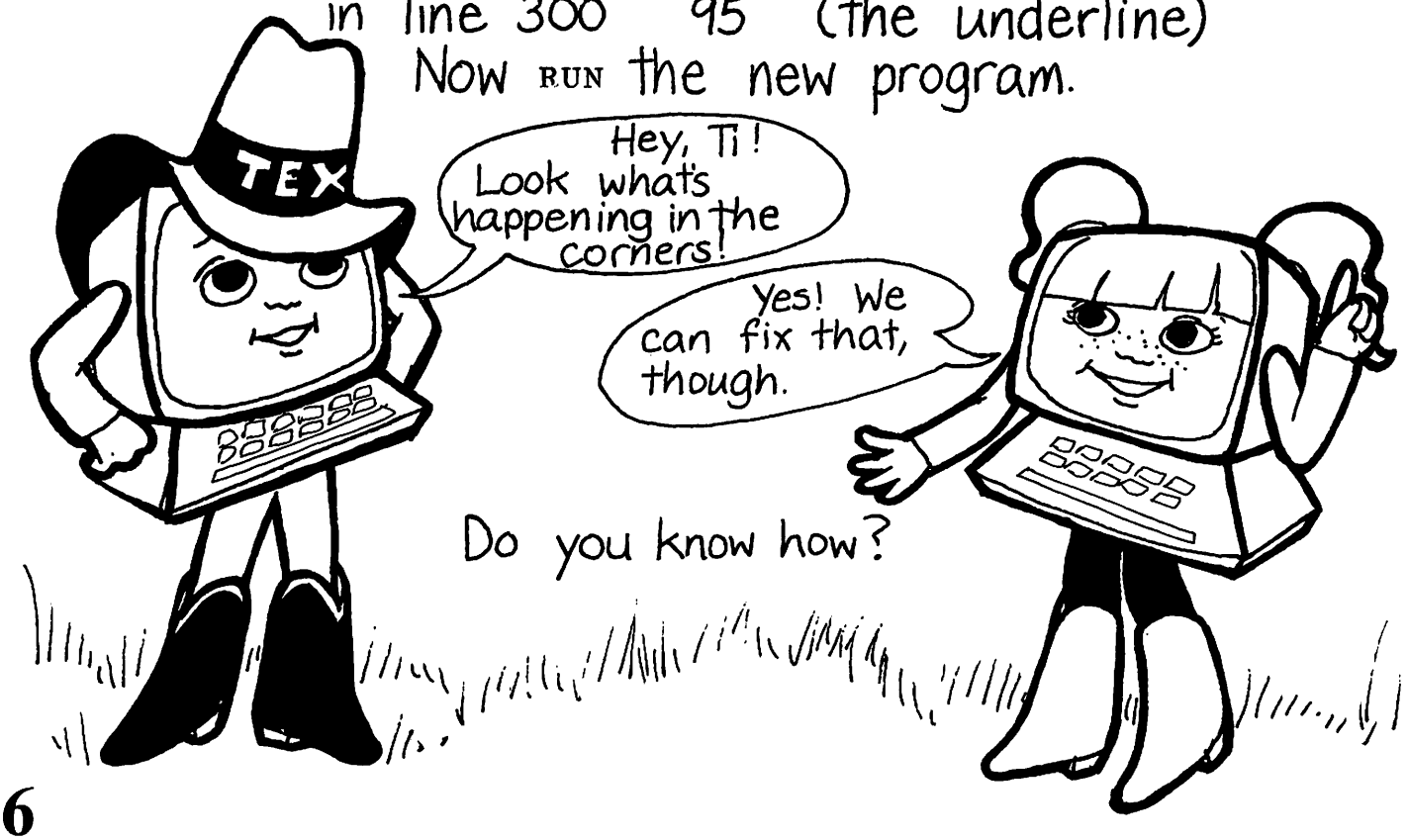
GRAPHICS CHARACTERS 24



So far in our graphics programs we have used the asterisk, but the TI-99/4A has many other available characters. Let's see some! Try entering the program we used in Lesson 21, GETTING FRAMED, but this time instead of 42, the ASCII code for *, let's use:

in	line 10	30	(the cursor)
in	line 100	45	(the minus sign)
in	line 200	33	(the exclamation point)
in	line 300	95	(the underline)

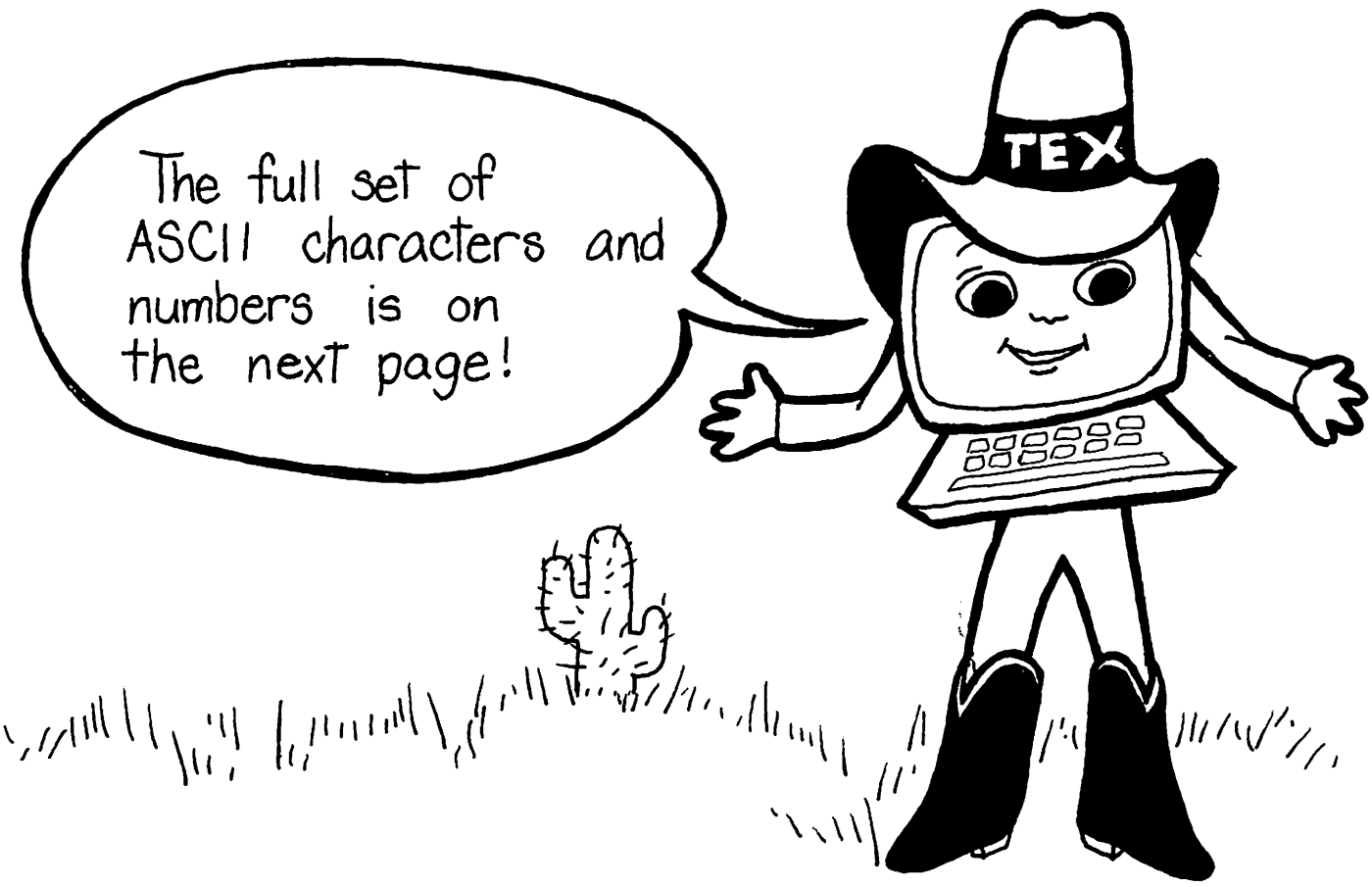
Now RUN the new program.



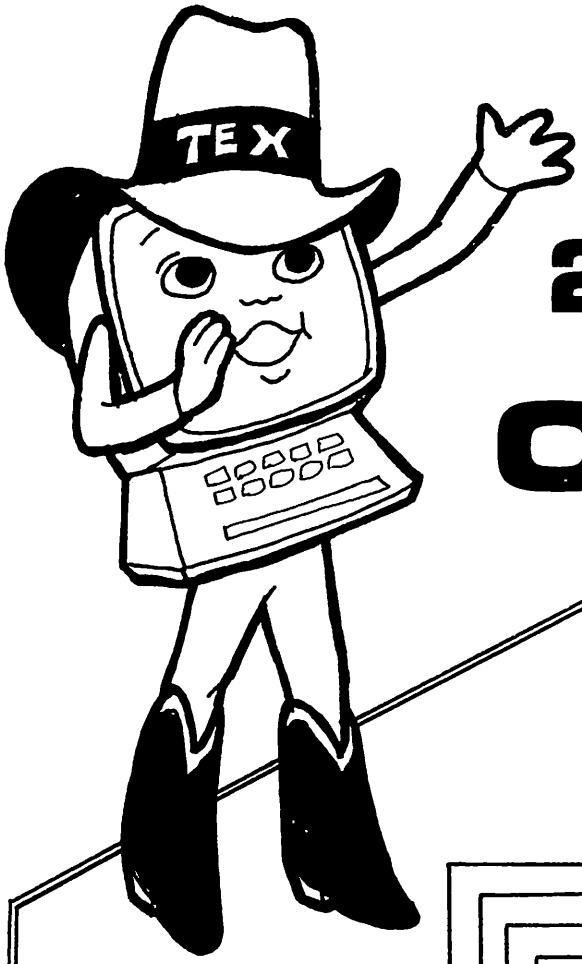
Do you know how?

If you said by shortening the lines, you're right! Here is one way:

```
10  CALL VCHAR (2,3,30,21)
100  CALL HCHAR (1,3,45,28)
200  CALL VCHAR (2,30,33,21)
300  CALL HCHAR (23,3,95,28)
```

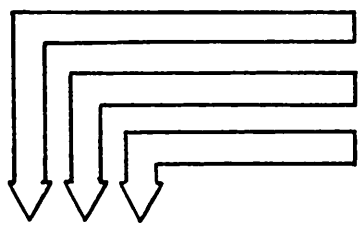


32		(space)
33	!	(exclamation point)
34	"	(quote)
35	#	(number sign)
36	$	(dollar)
37	%	(percent)
38	&	(ampersand)
39	'	(apostrophe)
40	(	(open parenthesis)
41	)	(close parenthesis)
42	*	(asterisk)
43	+	(plus)
44	,	(comma)
45	-	(minus)
46	.	(period)
47	/	(slant)
48	0	
49	1	
50	2	
51	3	
52	4	
53	5	
54	6	
55	7	
56	8	
57	9	
58	:	(colon)
59	;	(semicolon)
60	<	(less than)
61	=	(equals)
62	>	(greater than)
63	?	(question mark)
64	@	(at sign)
65	A	(taller letters)
⋮	⋮	
90	Z	
91	[	(open bracket)
92	\	(reverse slant)
93	]	(close bracket)
94	^	(exponentiation)
95	_	(underline)
96	`	(grave)
97	a	(shorter letters)
⋮	⋮	
122	z	
123	{	(left brace)
124	¦	
125	}	(right brace)
126	~	(tilde)
127	DEL	(appears on screen as a blank)



25.
call color

CALL COLOR is a command that gives you control of three sets of colors:



Screen
Foreground
Background

So, CALL COLOR (1, 2, 6) would give you a transparent screen with black foreground (letters) and a blue background.

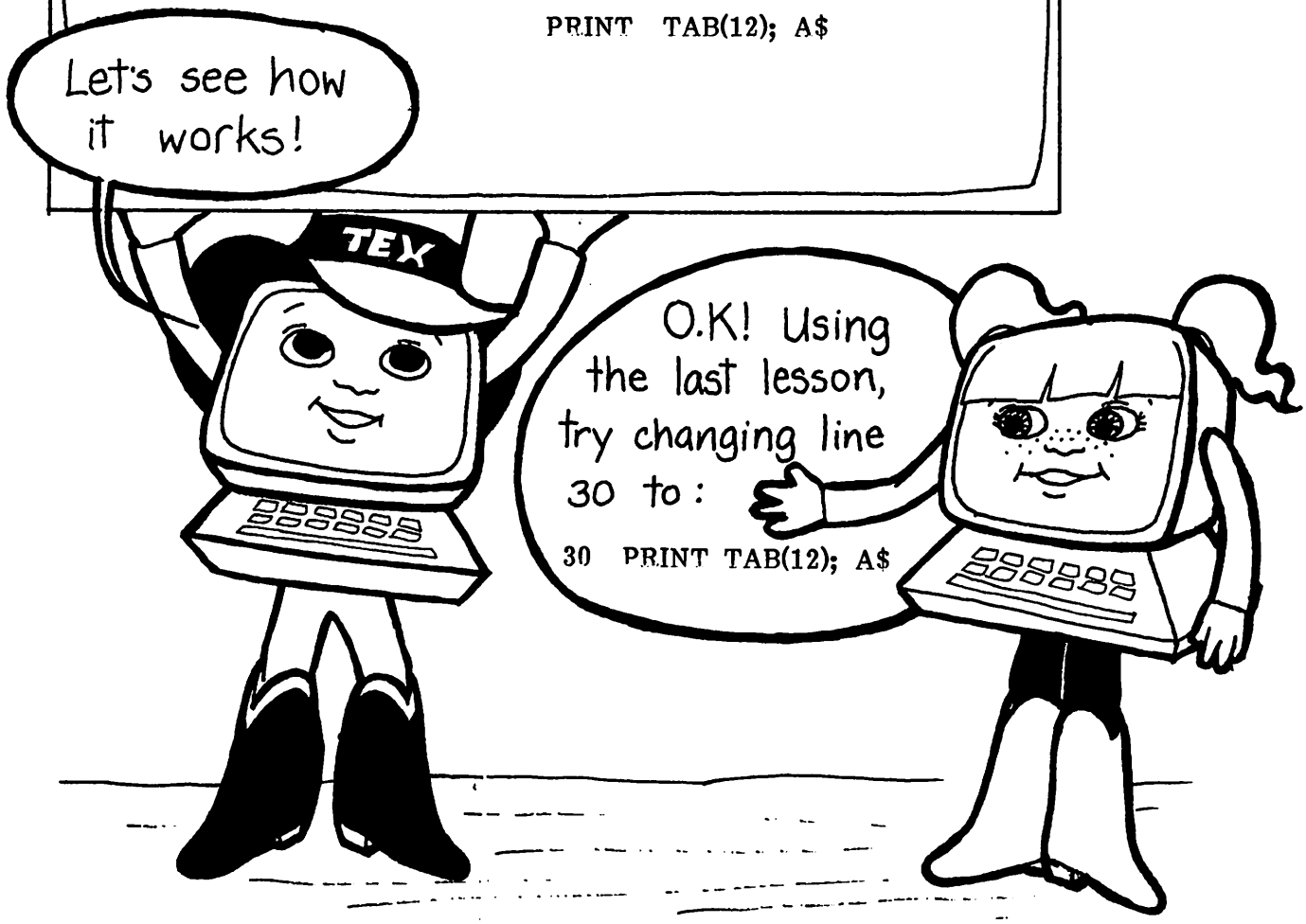
Try it so a friend can input a name.

```
 2  REM NAME IN LIGHTS
 5  CALL CLEAR
10  INPUT "YOUR NAME? ":A$
20  CALL CLEAR
30  PRINT A$
40  CALL COLOR (1,2,6)
50  GOTO 50
```

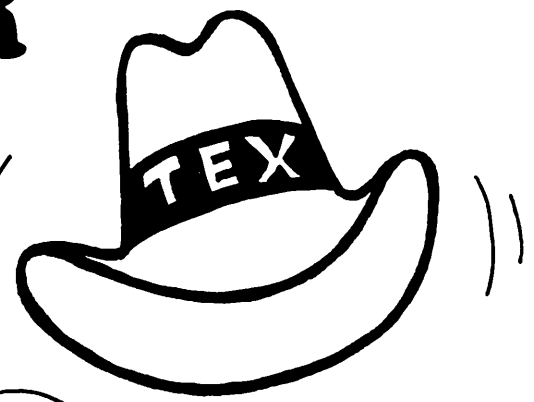
26. tab

TAB is a command which allows us to position words on the screen. You activate TAB by typing TAB with a number in parentheses, like this :

```
PRINT  TAB(12); A$
```



JUST FOR REVIEW

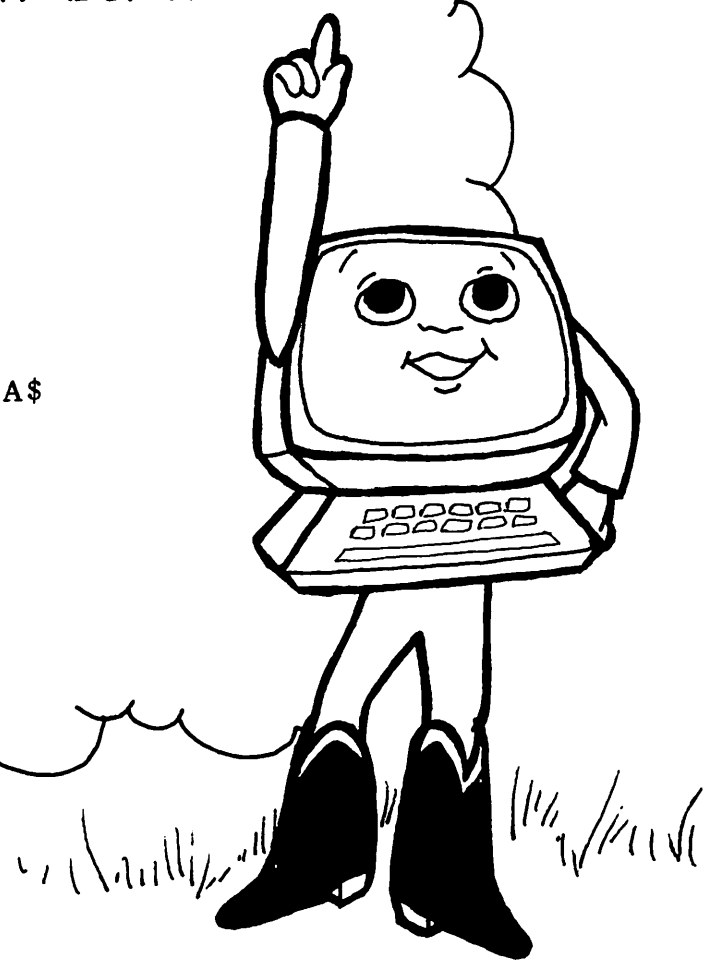


27.

Can you make your name climb up the center of a dark red screen with a light blue background, and give it a few seconds' delay between each line? Try it without looking at the program below.

Here is one way!

```
 2  REM CLIMBING NAME
 5  CALL CLEAR
10  INPUT "YOUR NAME? ":A$
20  CALL CLEAR
30  PRINT TAB(12); A$
35  CALL SCREEN (7)
40  CALL COLOR (1,2,6)
50  FOR X = 1 TO 1000
60  NEXT X
70  GOTO 30
```



GLOSSARY

CALL CHAR

- command to use numbers to define characters. 32 to 127 are ASCII, and 128 to 159 are available for additional characters.

CALL CLEAR

- command to clear the screen.

CALL COLOR

- command to control the colors in a program.

CALL HCHAR

- command to control horizontal characters.

CALL SCREEN

- command which enables the user to change screen color.

CALL SOUND

- command which enables the user to use the sound capabilities of the computer.

CALL VCHAR

- command to control vertical characters.

CHARACTER

- any one letter, number (0-9), space, or symbol.

COMMAND

- a specific order to the computer, such as RUN, PRINT, LIST, CONTINUE, etc.

CONTINUE

- if you stop a program by holding down the FUNCTION key and hitting 4, you can begin it again by typing CONTINUE and pressing the ENTER key.

CURSOR

- the blinking black rectangle that moves on the screen. It is where the next character will be.

DATA

- a list of values to be given to the READ statements.

EDIT

- command used with a line number to call up a line in a program to be edited or changed.

EDITING MODE

- mode in which you are able to alter, delete, add to, or otherwise edit the statements in a program.

ENTER

- key that enters whatever you have typed on the screen into the computer's memory.

EXECUTION MODE

- mode in which the computer executes the command you have given in the program.

FOR-NEXT

- the "For" statement tells the computer to do something a certain number of times. The "Next" statement comes after the "For" statement and tells the computer to go back and do the action stated. The "For-Next" statement is a type of loop. Example:

```
10 FOR A = 1 to 15
15 PRINT A
20 NEXT A
```

FUNCTION KEY (abbreviated FCTN)

- when held down allows other keys to function as marked on their sides.

FCTN A :
C \
D → moves the cursor forward one space without changing what was in that space
E ↑ in editing, moves cursor to line above
F { (left brace)
G } (right brace)
I ? (question mark)
O ' (apostrophe)
P " (quotation marks)
R [(left bracket)
S ← backspace without removing what was there before
T] (right bracket)
U _ (underline)
W ~ (tilde)
X ↓ in editing, moves cursor to line below
Z \

FCTN-1 DELETE—allows you to delete material to the right of the cursor one space at a time.
2 INSERT—allows you to insert material into an already entered line.
3 ERASE—allows you to erase an entire line.
4 CLEAR—will break a running program. The same key may be used when entering a program; in this case it will scroll the screen up one line without entering that line.
FCTN- ± QUIT—will take the computer out of BASIC.

GOSUB (or GO SUB)

- command used to send the computer to a subroutine - it is always used with the RETURN command.

GOTO

- tells the program to go to the line number stated; the program continues from that line. Example: GOTO 100.

IF-THEN

- a program device for the computer to go to another part of the program if a relationship is true. For example:

```
20 IF A<>B THEN GOTO 70
30 IF A = B THEN 40
40 PRINT B: END
70 PRINT
```

IMMEDIATE MODE

- the computer is in the immediate mode when it has a command or a list of commands which are to be executed as soon as RETURN is pushed. In the immediate mode the commands do not have line numbers, which are necessary for the program mode.

INPUT

- enables the program user to type in a response while the program is running. Example:

```
10 INPUT A
RUN
? 5
```

INT (or INTEGER)

- a command to tell the computer to round off to a whole number or integer.

LET

- statement setting a value, as in LET A = 13.

LINE NUMBER

- a number typed at the beginning of a memory line.

LIST

- shows everything that is in the computer's memory in line number order.

MEMORY

- where information is stored inside the computer. There are two kinds of memory: RAM and ROM. RAM means RANDOM ACCESS MEMORY and is used for the material you enter. ROM is READ ONLY MEMORY and is used by the computer manufacturer to store languages such as BASIC—and other material that you need to know.

MENU

- a list of numbered choices in a program. It asks the program user to select one.

NEW

- command that erases everything in the random access memory.

OUTPUT

- what you see on the screen, printer, or other hardware.

PRINT

- a command that tells the computer to write on the screen whatever follows PRINT. Example: PRINT 2 * 5
  10

PROGRAM

- instructions to the computer telling it what to do and how to do it.

PROGRAM MODE

- mode in which computer stores in its memory instructions and their order of execution—as in:

```
10 INPUT A$
20 PRINT A$
```

RANDOM (RND(1))

- a command that tells the computer to choose a random number. Must appear after the word RANDOMIZE.

RANDOMIZE

- commands the computer to turn on Random number function.

READ

- puts value from DATA statements to variable(s). Ex.: 100 READ A, B$

REM (or REMARK)

- REM allows the programmer to make notations in the program without having those notations become part of the program.

RETURN

- command at the end of a subroutine which brings the computer back to the next line of the program after that which told it to go to the subroutine.

RUN

- a command that starts the program executing.

STRING

- a group of characters represented by a two or more digit symbol ending in a $—as in A$

TIMER

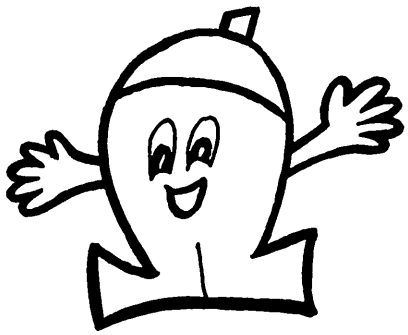
- delays a program. It leaves a certain amount of time between one part of a program and another point.

TRACE

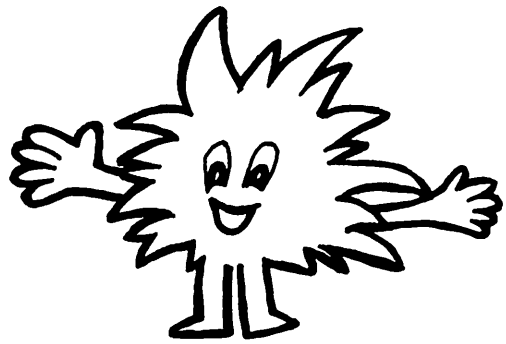
- command which, when used with RUN, will have the computer print on the screen all the steps it goes through in a program in the order that it does them. Must be used with RUN.

UNTRACE

- command which turns off TRACE.



Hi!



I'm Chip! I'm Corey!

When you finish with this book,
you will be ready for us.
We'll take you all the way on the
road to **Computer Literacy!**

We'll introduce you to. . .

History of Computers
Types of Computers
Uses of Computers
Parts of Computers
ASCII Code
Bits & Bytes
Computer Puzzles
Computer Memory
Crossword Puzzles
Binary Number System
Computer Languages
Programming
Keyboarding
Flow Charting
Graphics
BASIC Dialects
Computer Ethics
Serial & Parallel Circuitry

INCLUDES COMPLETE PROGRAMMING SECTION

You'll find us in

THE COMPUTER PRIMER

by Ann Cavanaugh

481 pages
89 photographs
Hundreds of illustrations
$12.95

APPROVED for NYSTL--#0898240468
New York City Board of Education Approved--Contract #6282300

TRILLIUM PRESS
ISBN: 0-89824-059-X